Good morning!

I had a sore throat and
I lost my voice…

Seems to happen once a year.

So this will be a “silent” lecture.

Grammar of Graphics

with R & ggplot2

SP-2014 Lecture 20

Exercise

If you have not yet installed R,
start downloading R

http://stat.ethz.ch/CRAN/

Exercise

Start R

Download & install ggplot2 package

```
install.packages('ggplot2')
```

Load ggplot2 package

```
library('ggplot2')
```

ggplot2's "diamonds" data frame

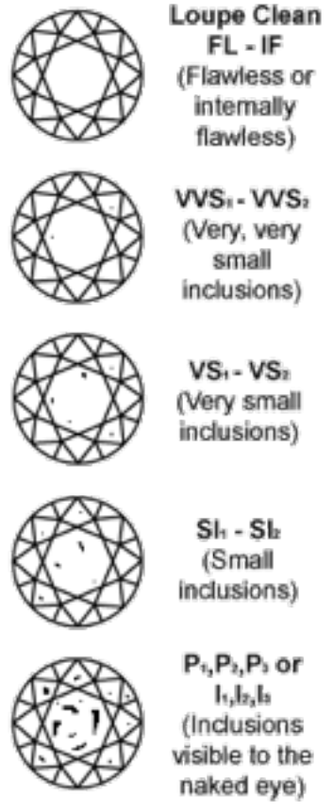
ggplot2's **diamonds** data frame

carat vector	**price** vector	**cut** factor	**color** factor	**clarity** factor
0.9	1200	Fair	G	VS1
1.2	5000	Good	E	SI2
...	...	...	...	...

10 columns

53940 rows

clarity



cut

color

Grammar of Graphics

ggplot

Grammar of Graphics

In ggplot2, plots are defined by this "grammar" (in EBNF)

plot ::= coord scale+ facet? layer+

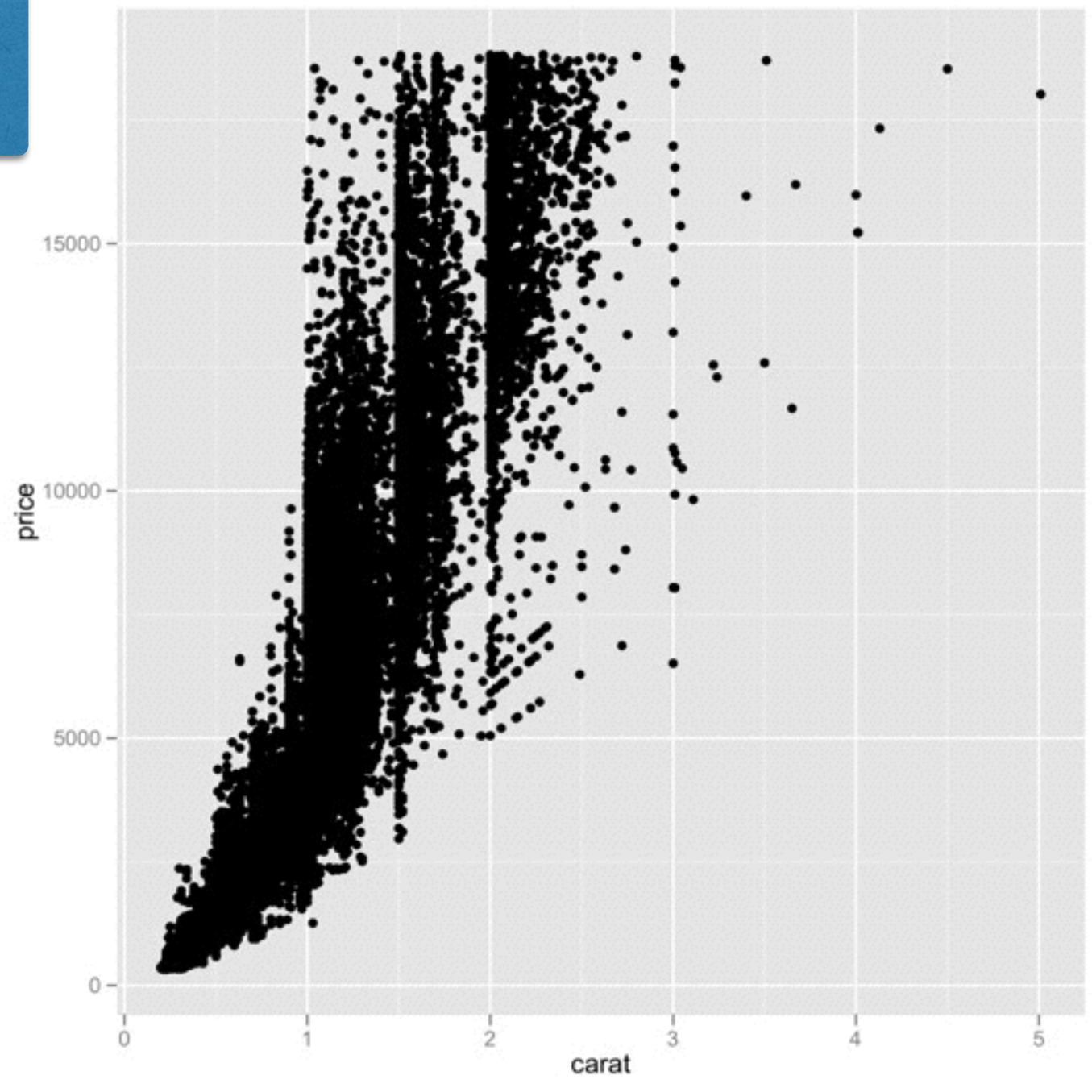
layer ::= data mapping stat geom position?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Here's an example

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

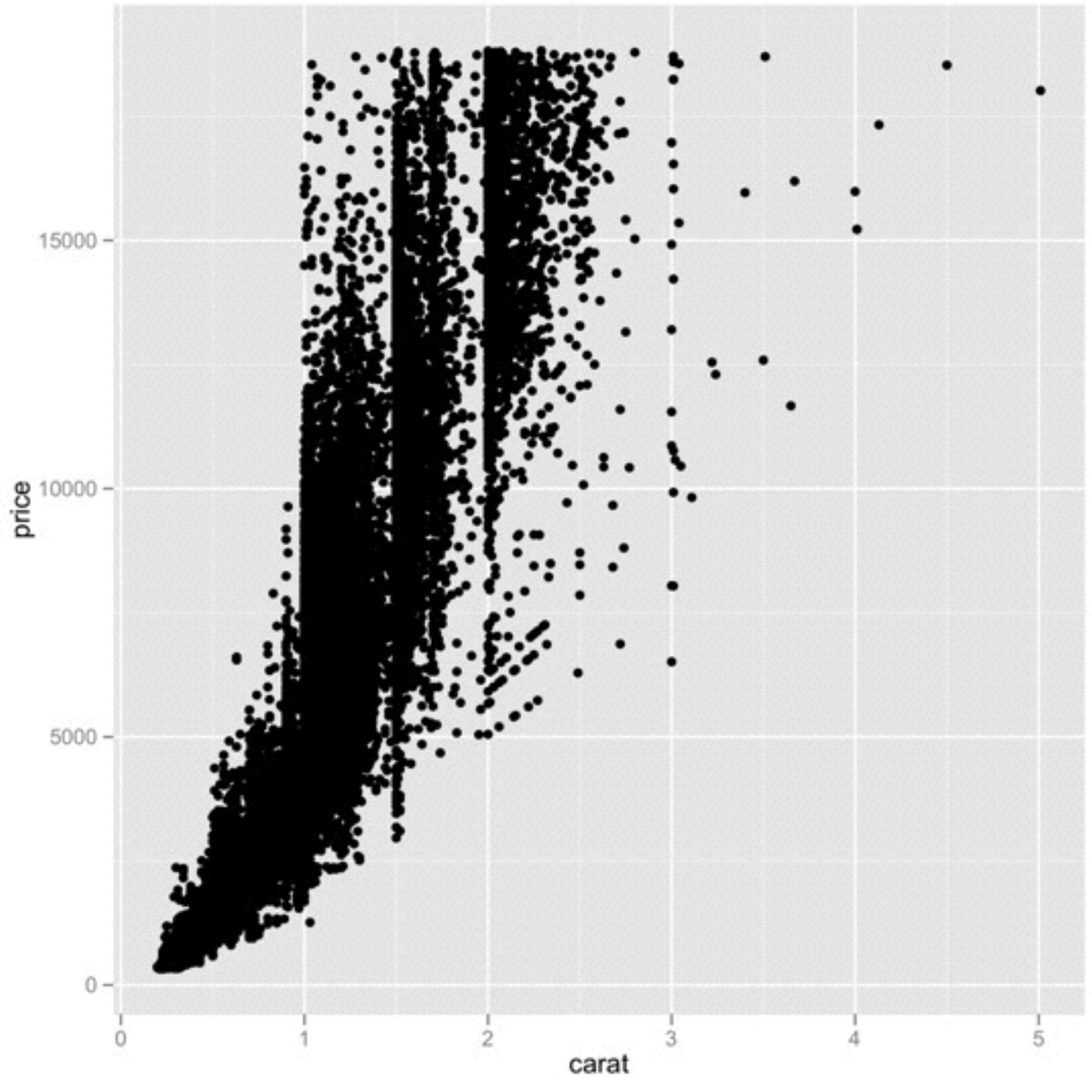


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Create a plot

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

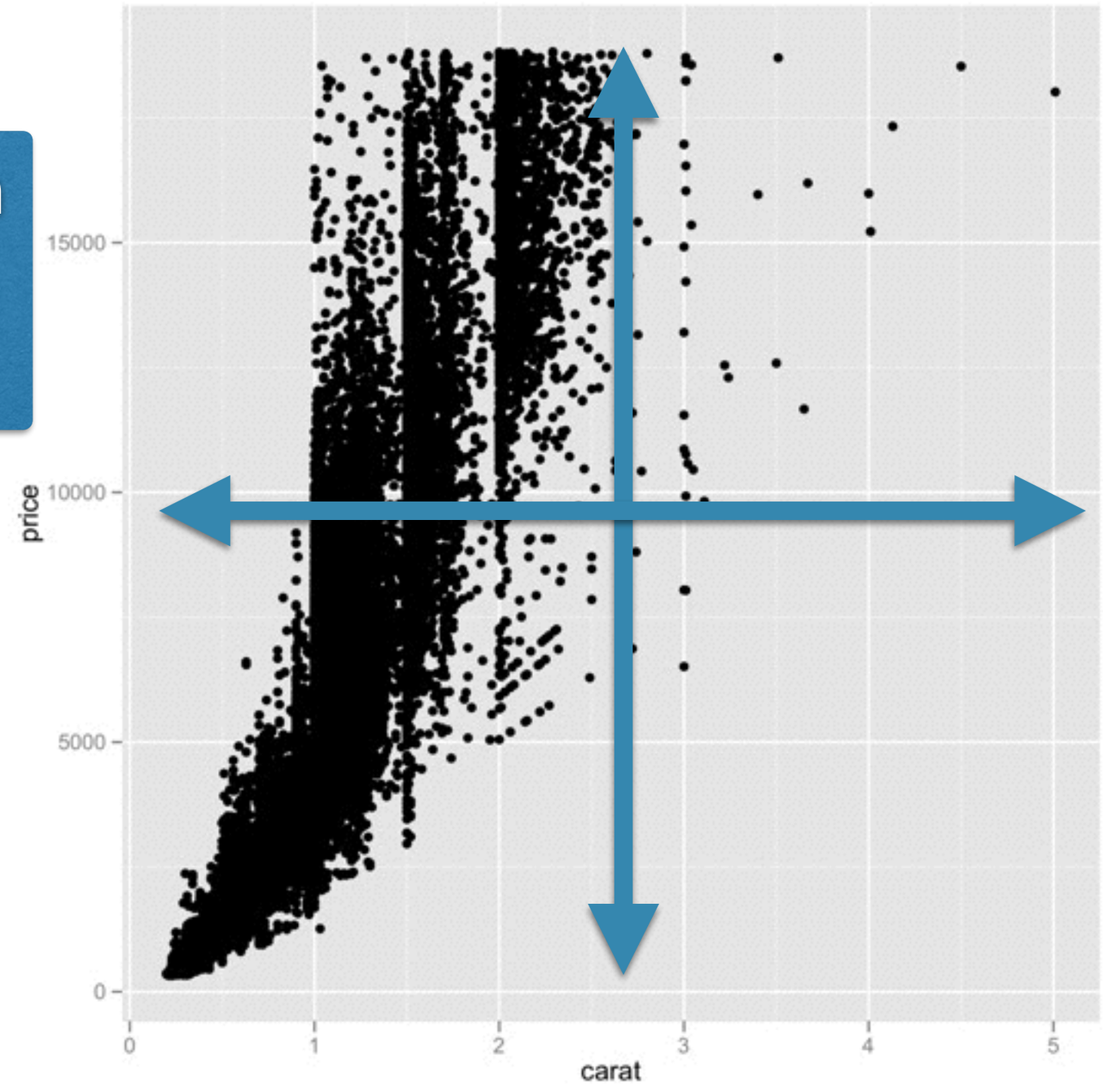


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous(
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

use a cartesian coordinate system

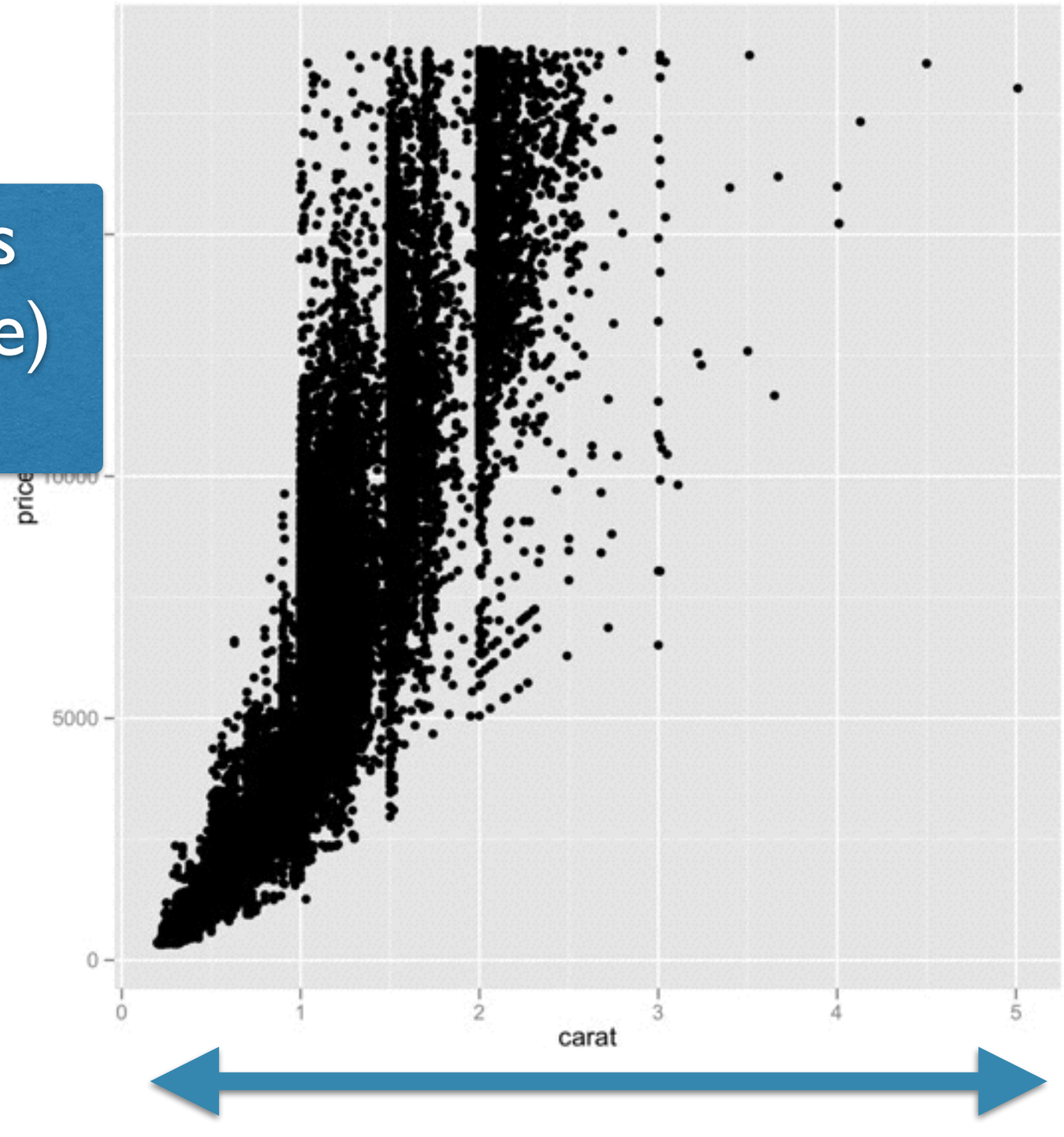


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

continuous (quantitative) x-axis

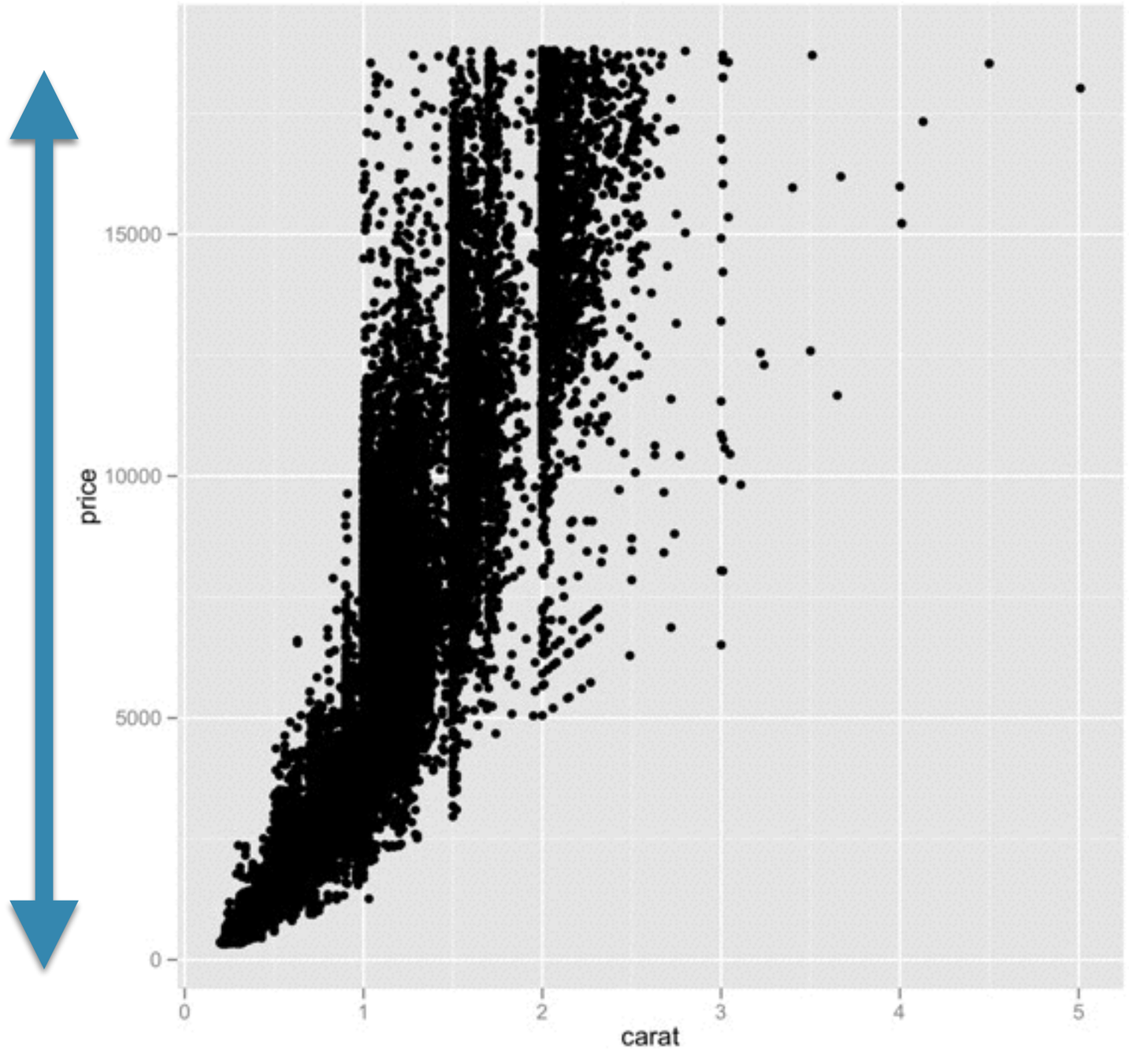


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

continuous (quantitative) y-axis

```
ggp
co
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

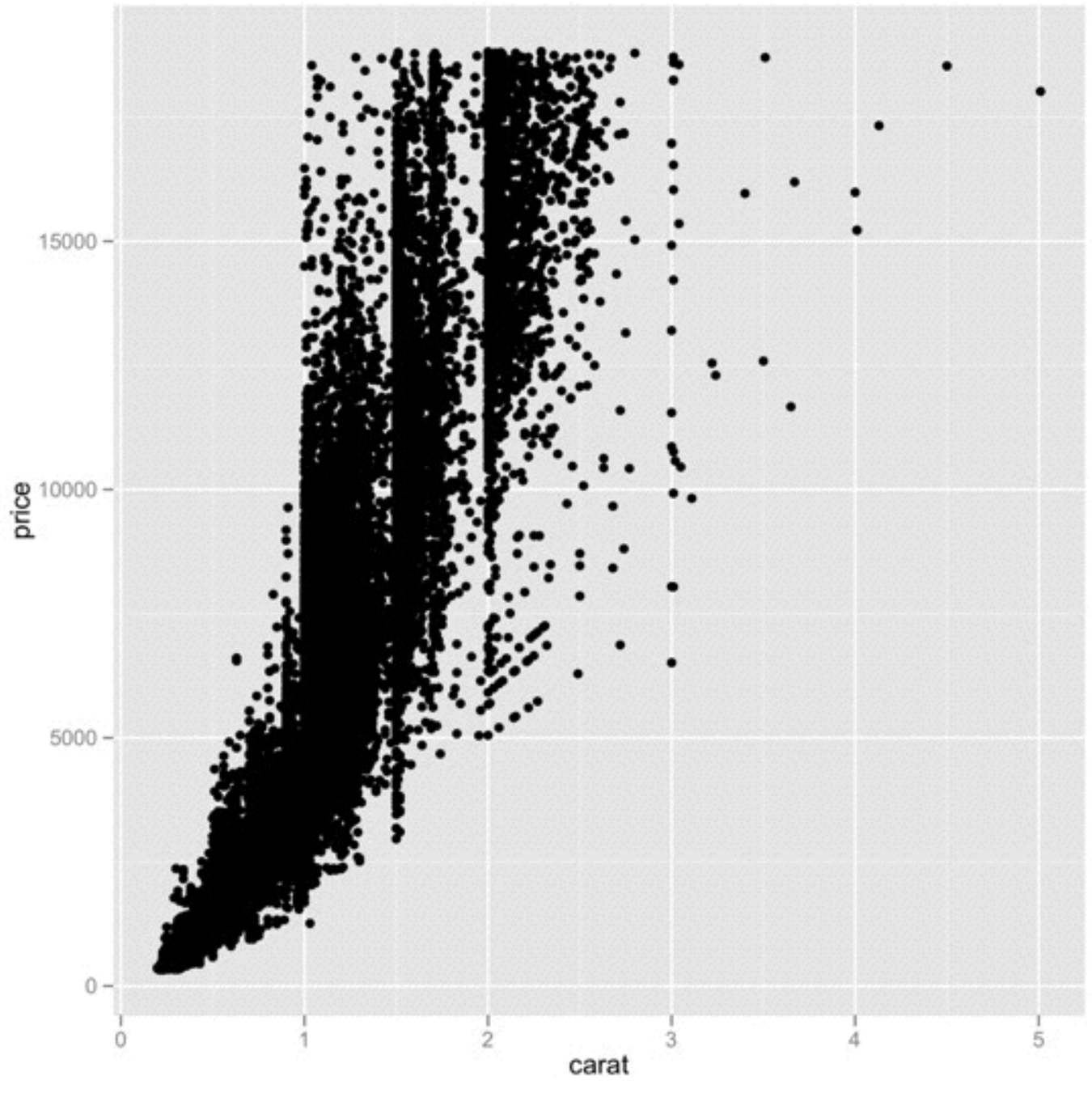


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
co
sc
sc
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

use diamonds data.frame

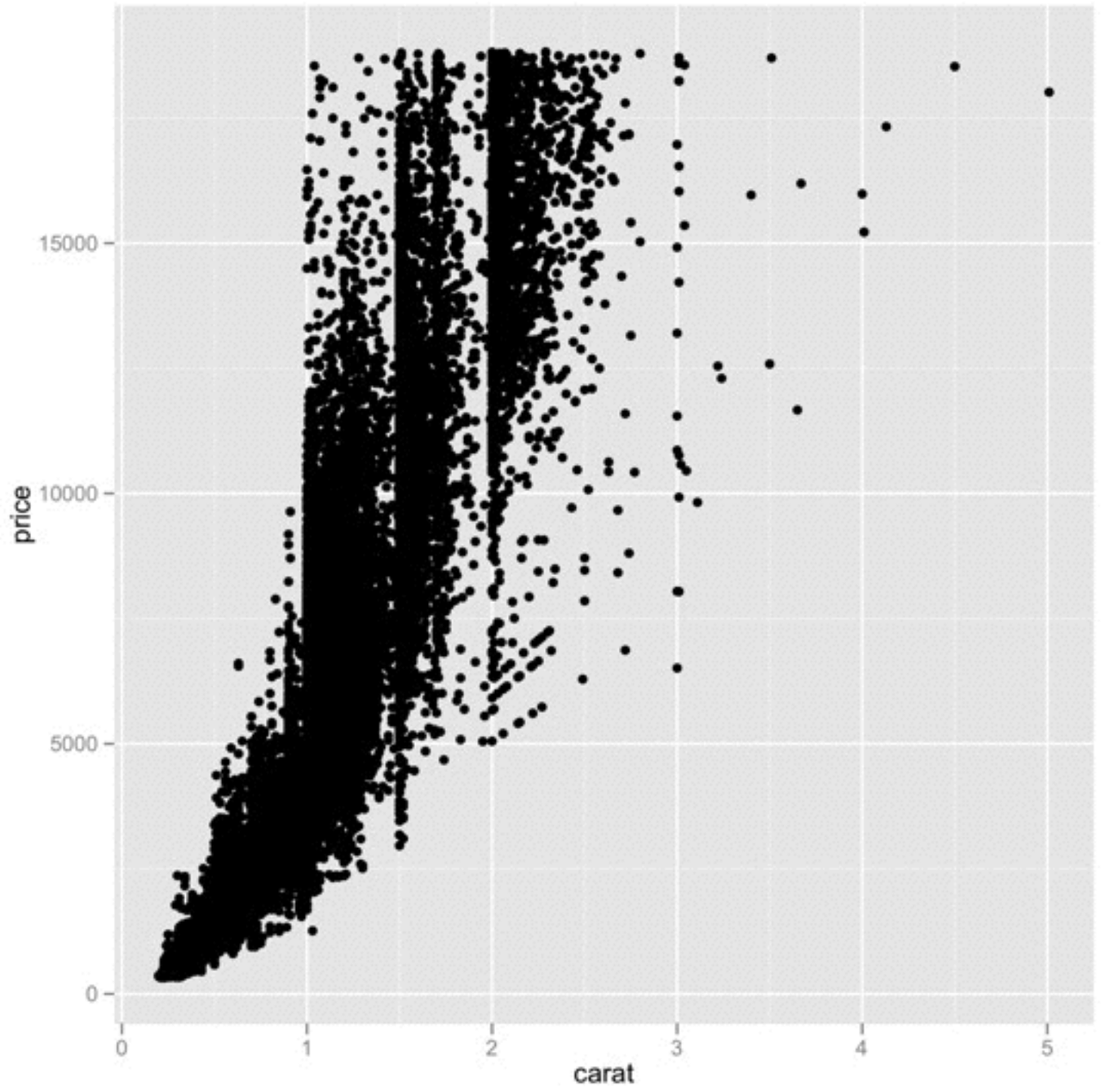


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot()
coord_
scale_
scale_
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

how to map data to visual "aesthetics"

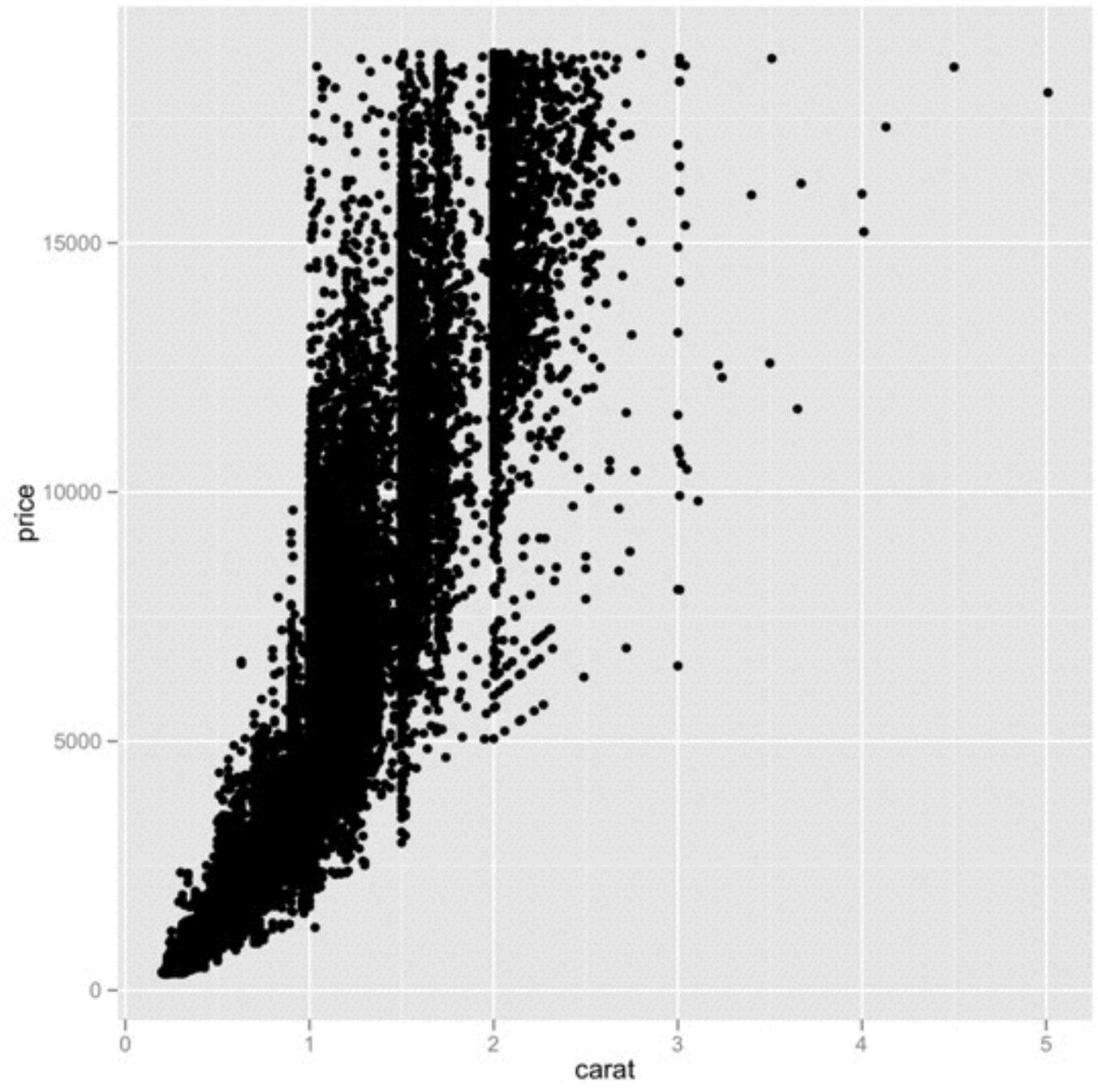


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartes
scale_x_cont
scale_y_cont
layer(
  data=diamonds
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

map diamonds$carat to x-axis

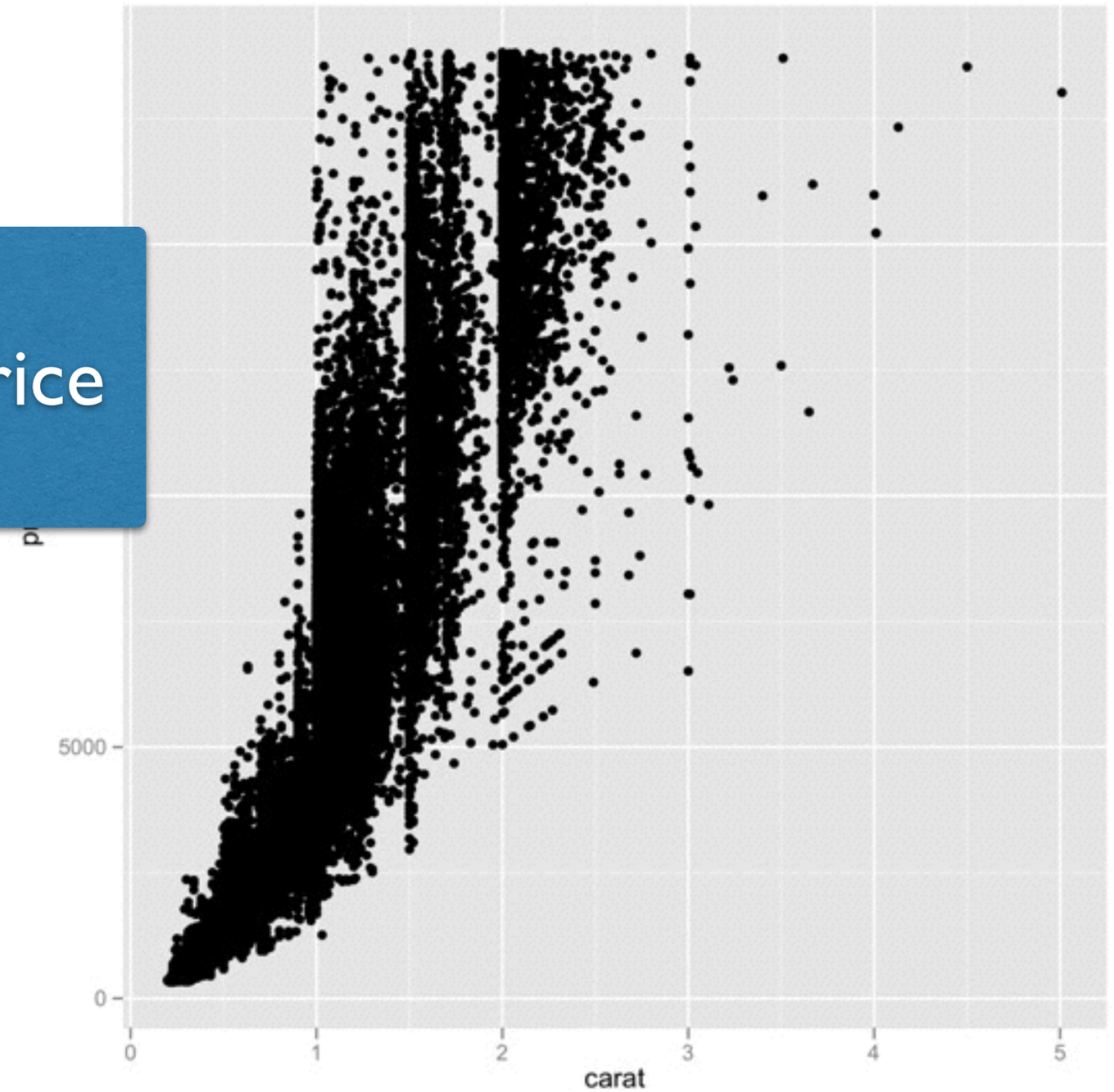


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

map diamonds$price to y-axis

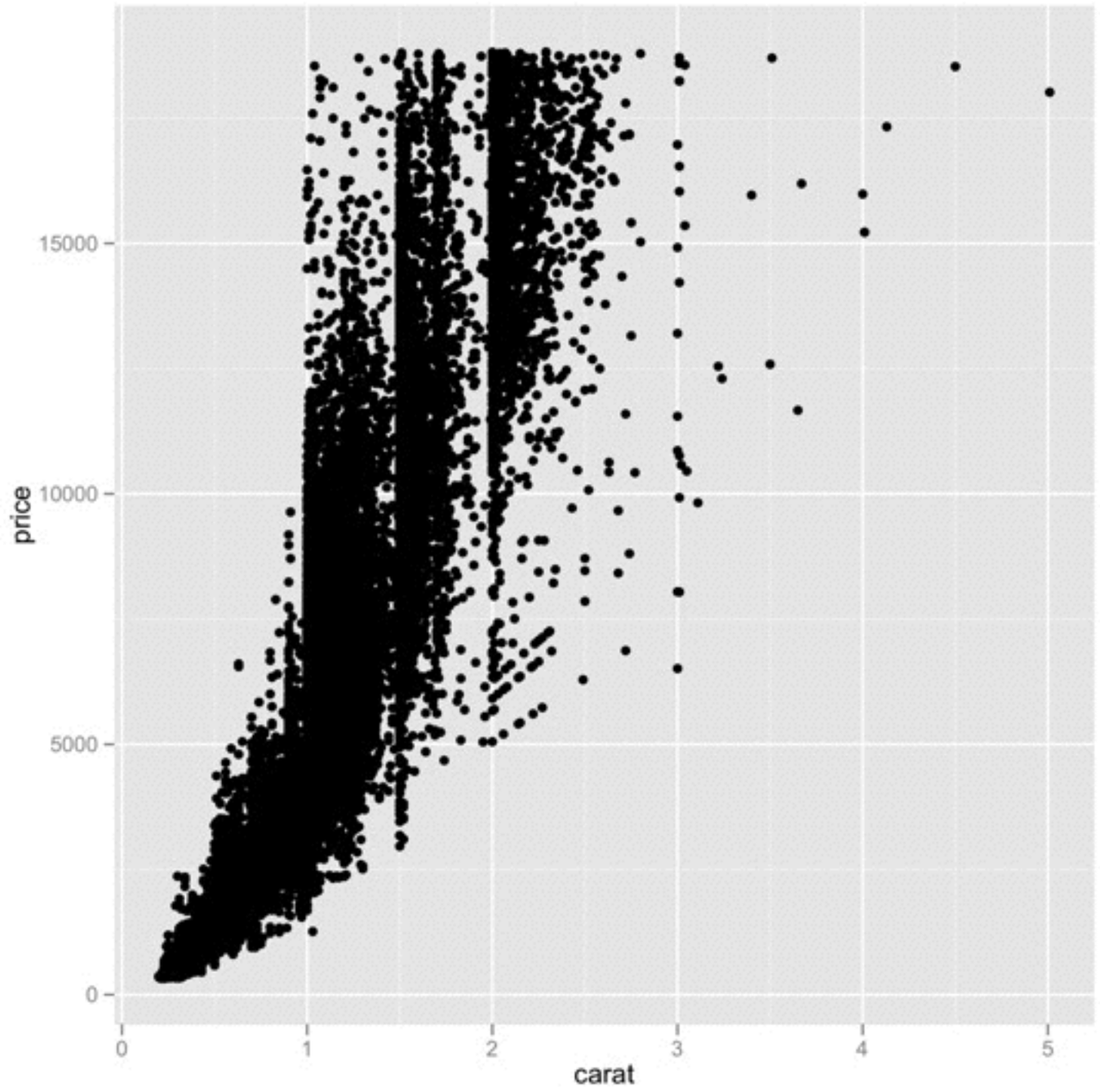


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
c
s
s
la
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

use each observation (row of diamonds) as is (no statistics)

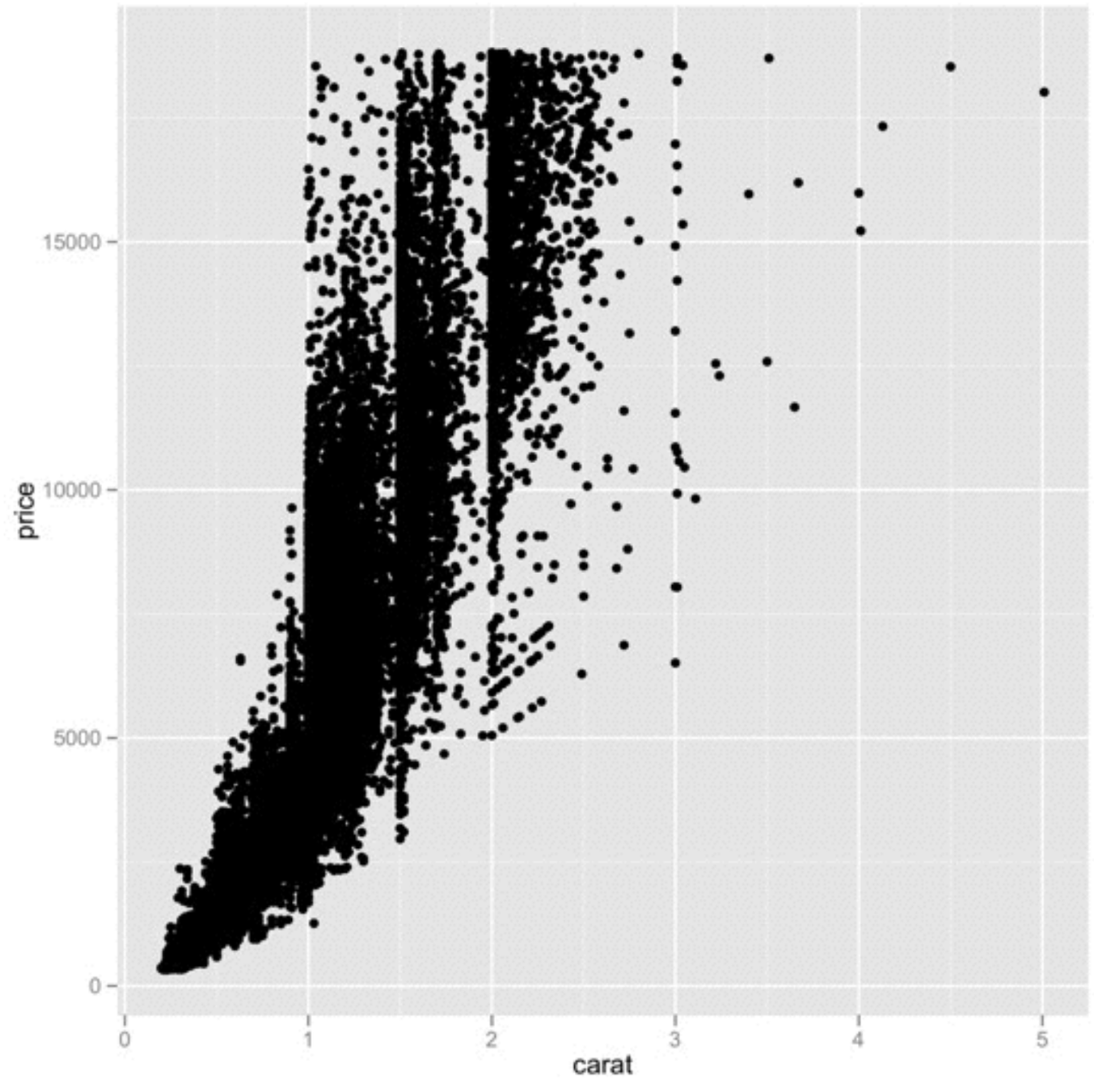


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds, mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```

draw one point for each observation

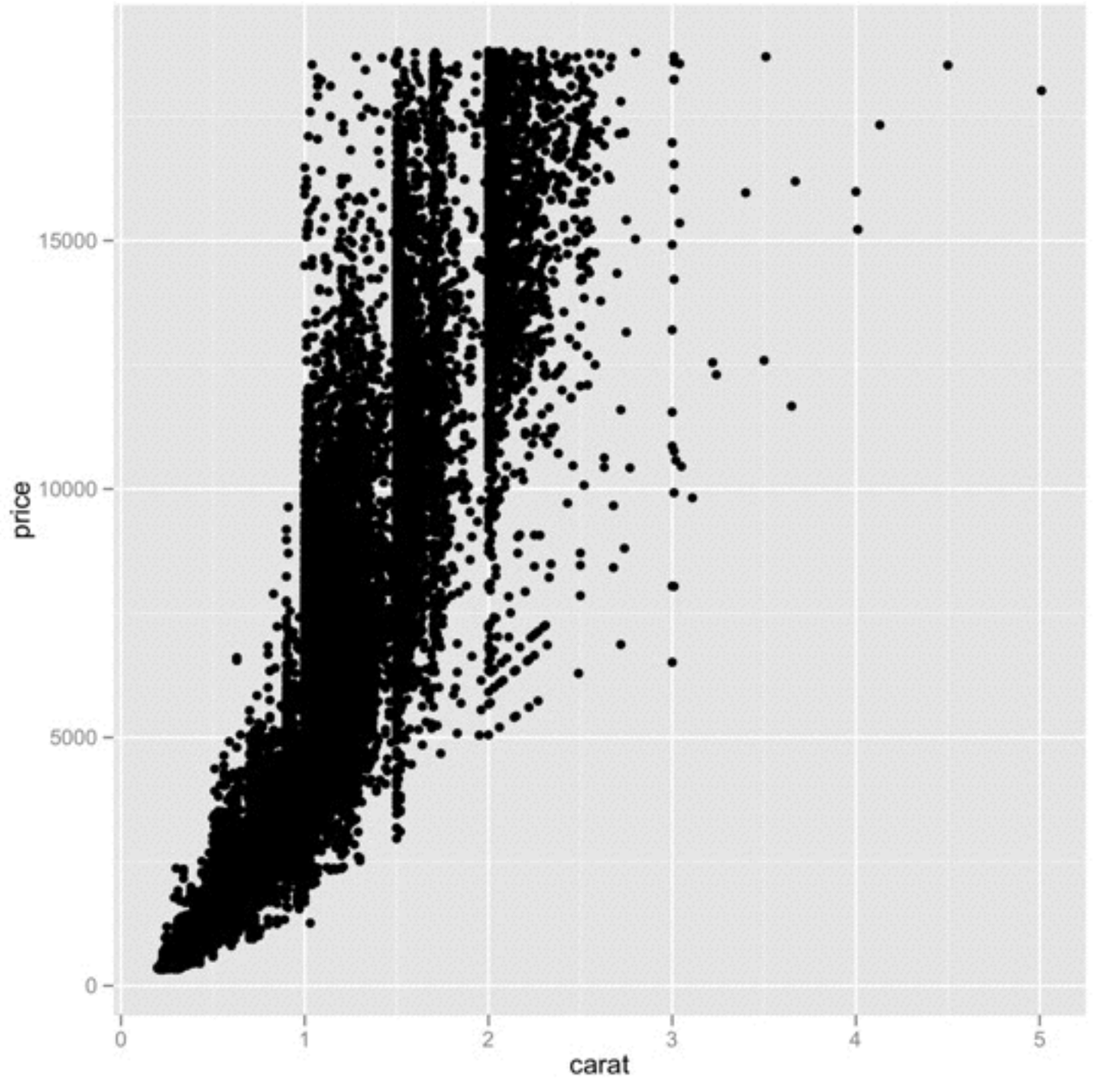


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=
  mapp
  stat=
  geom="point", geom_params=list(),
  position=position_identity()
)
```

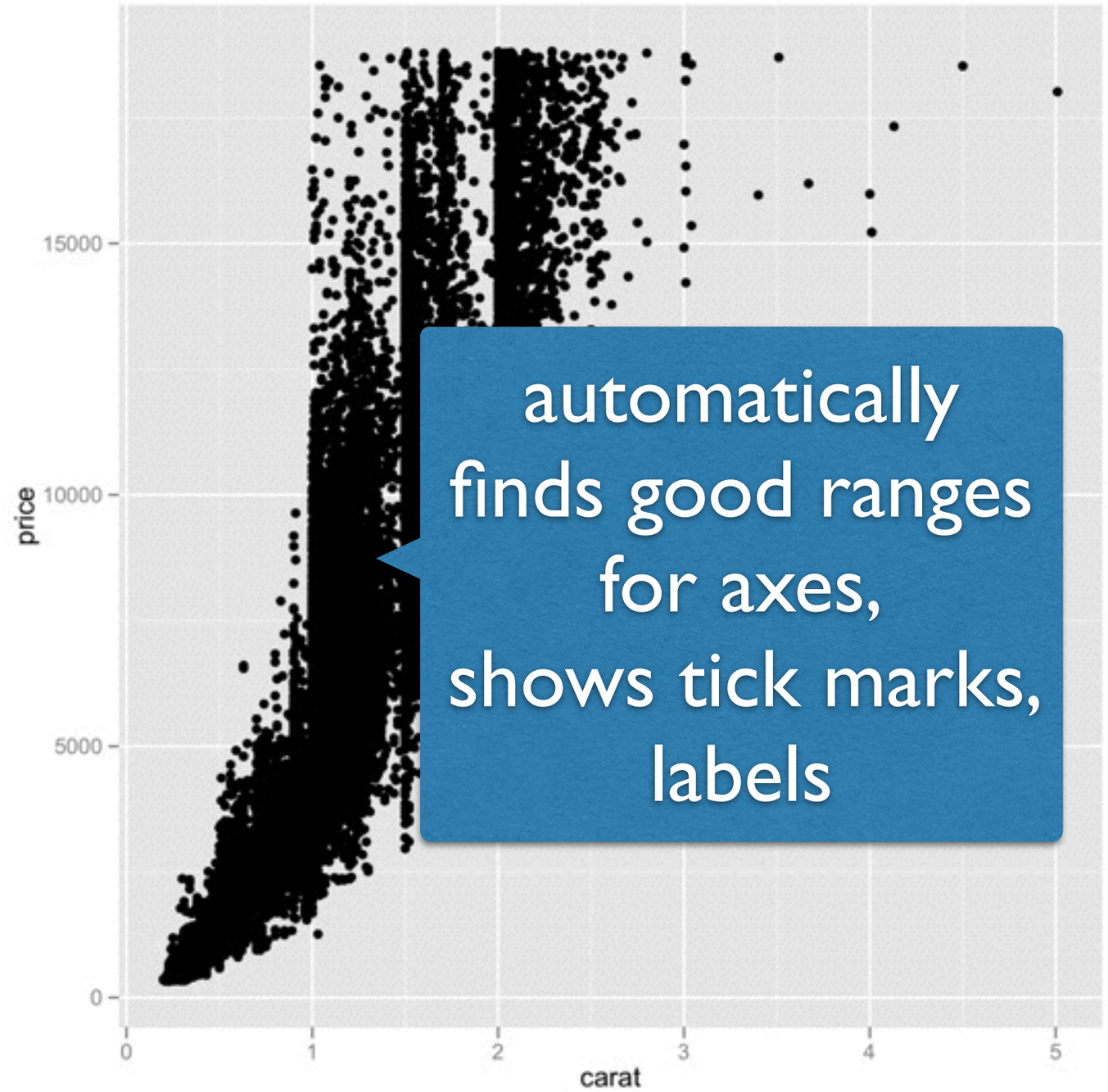
don't "adjust" the x or y position



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)
```



plot ::= coord scale+ facet? layer+

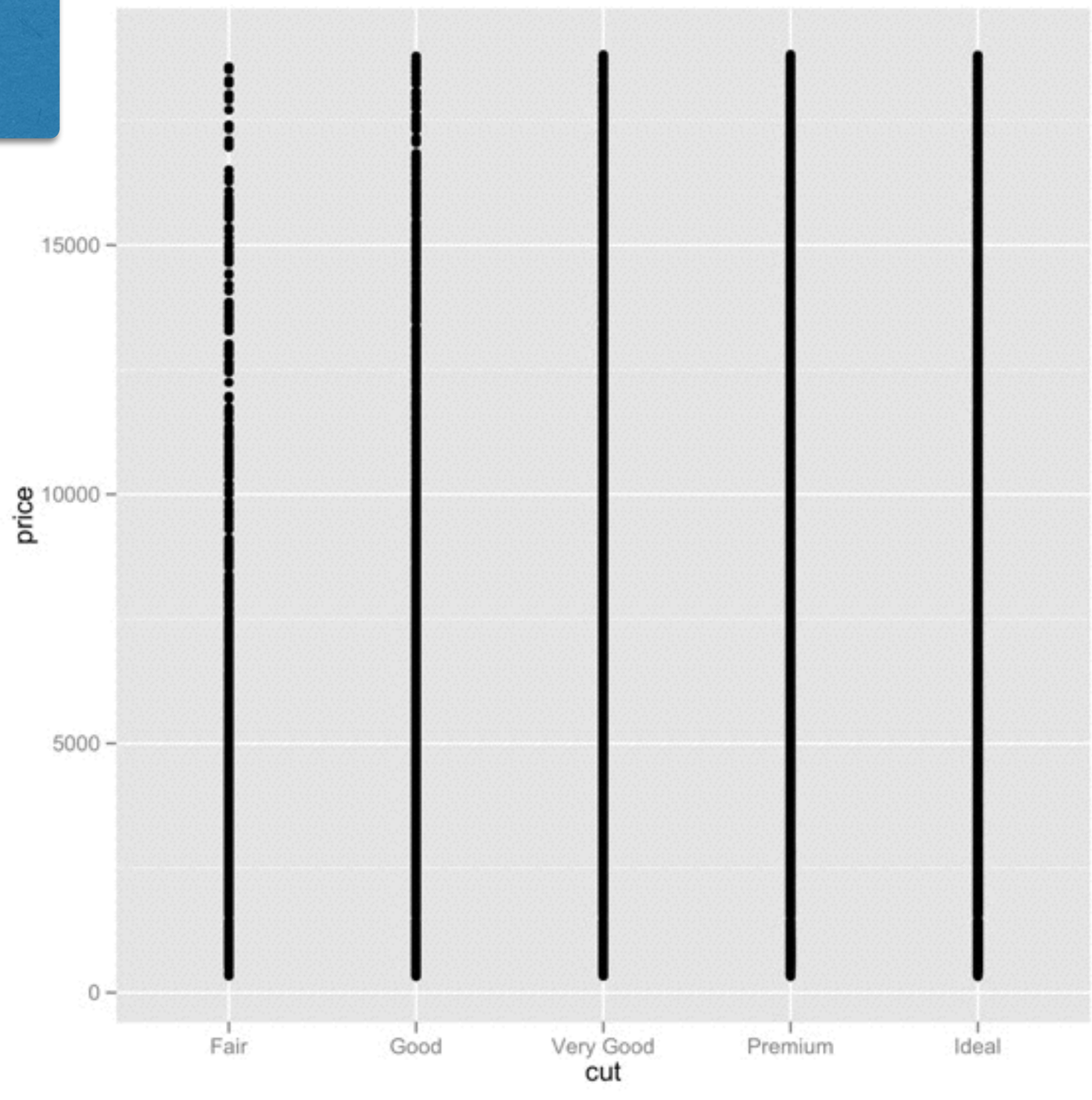
layer ::= data mapping stat geom position?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Another example…

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

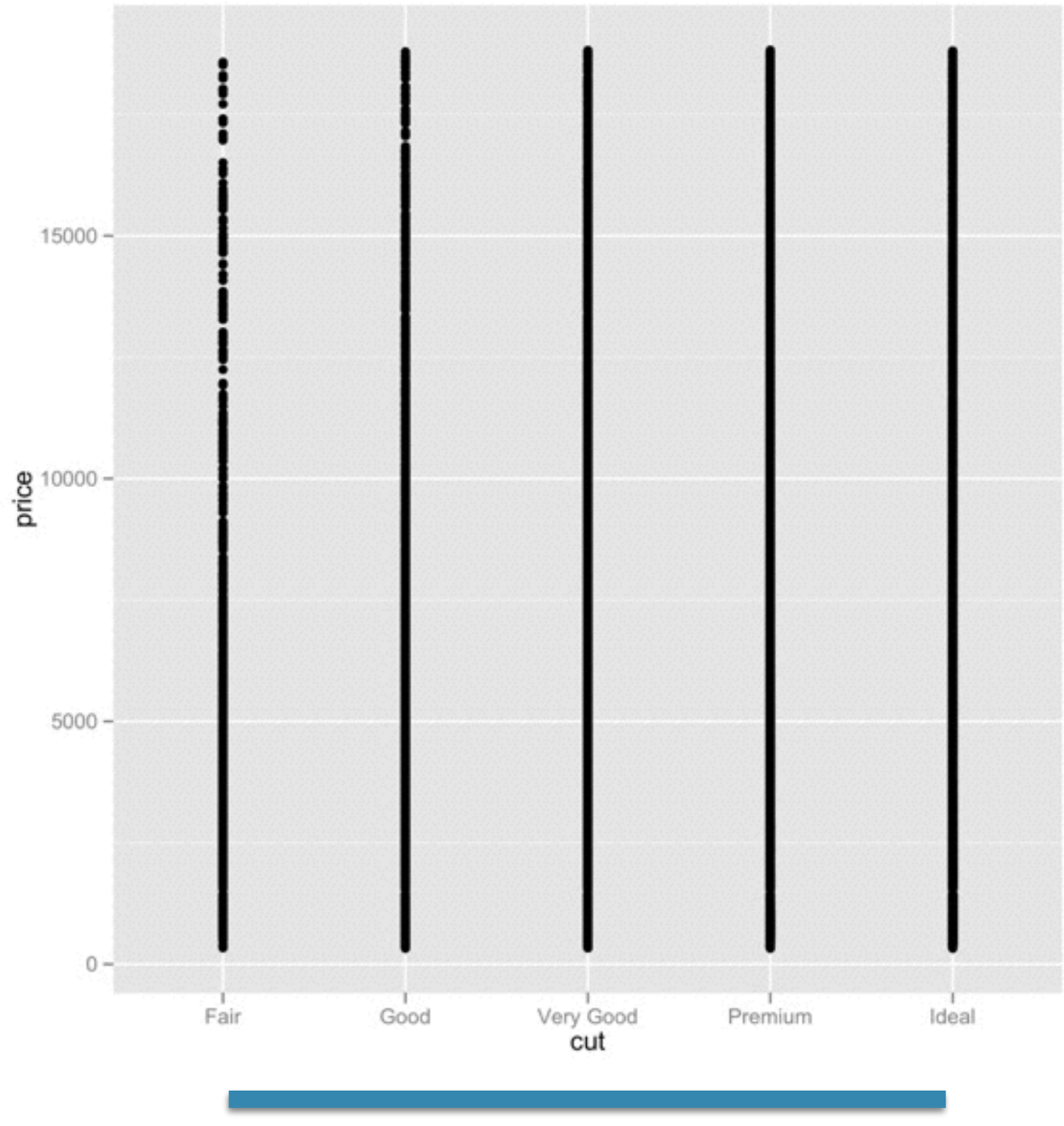


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

use a discrete scale for x-axis

```
ggp
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

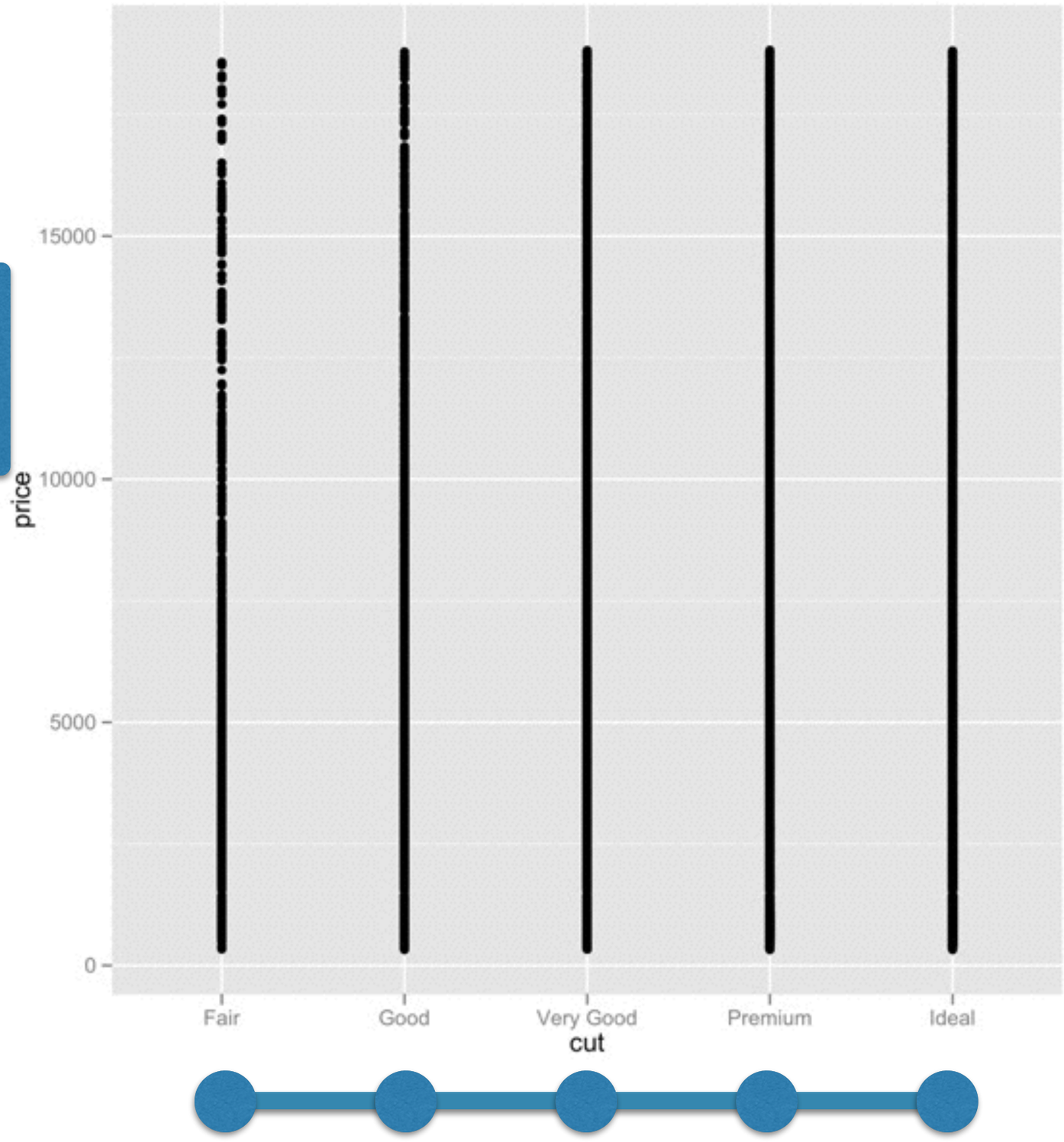


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesi
scale_x_discr
scale_y_conti
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

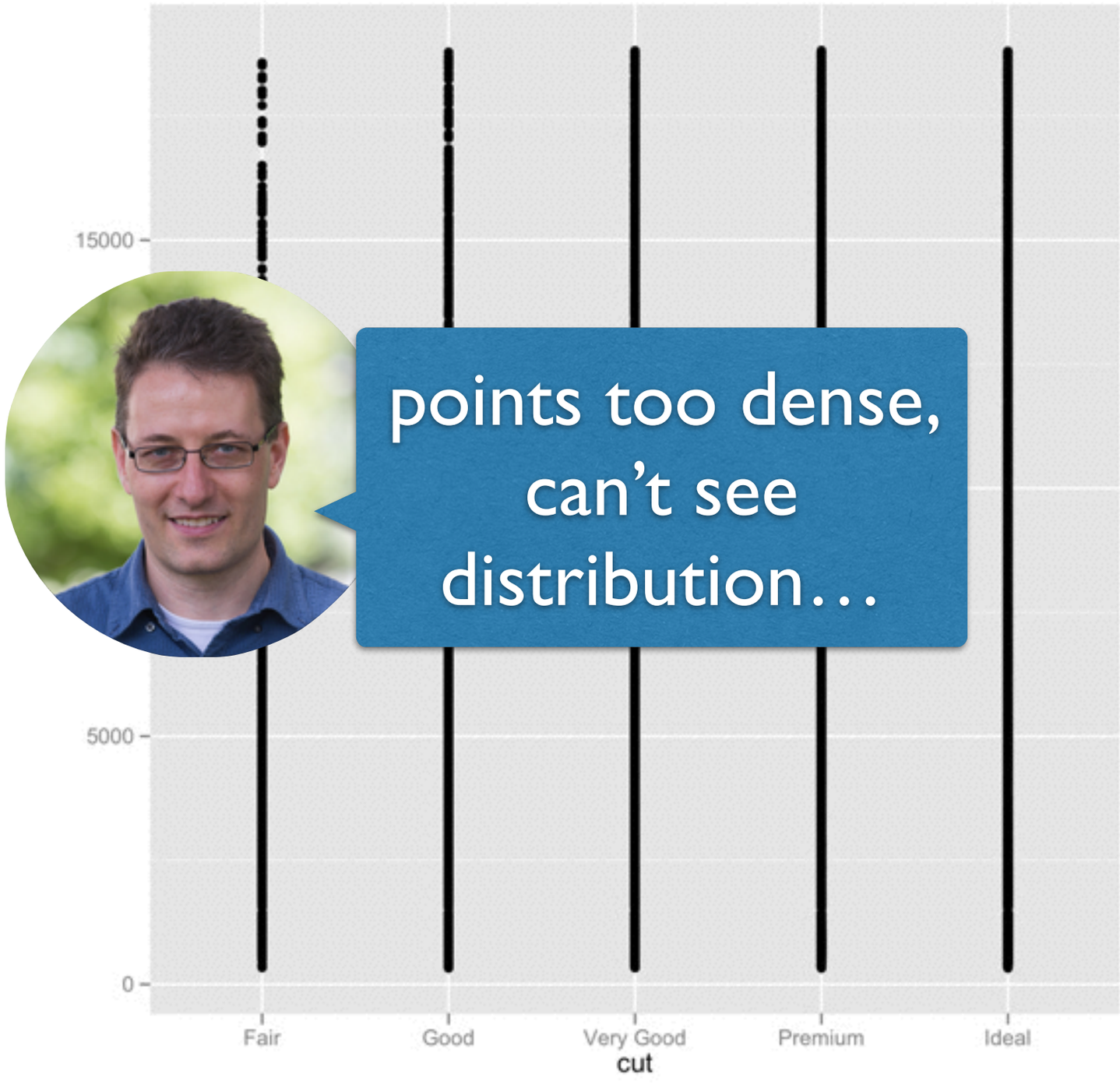
cut is a factor (with 5 levels)



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

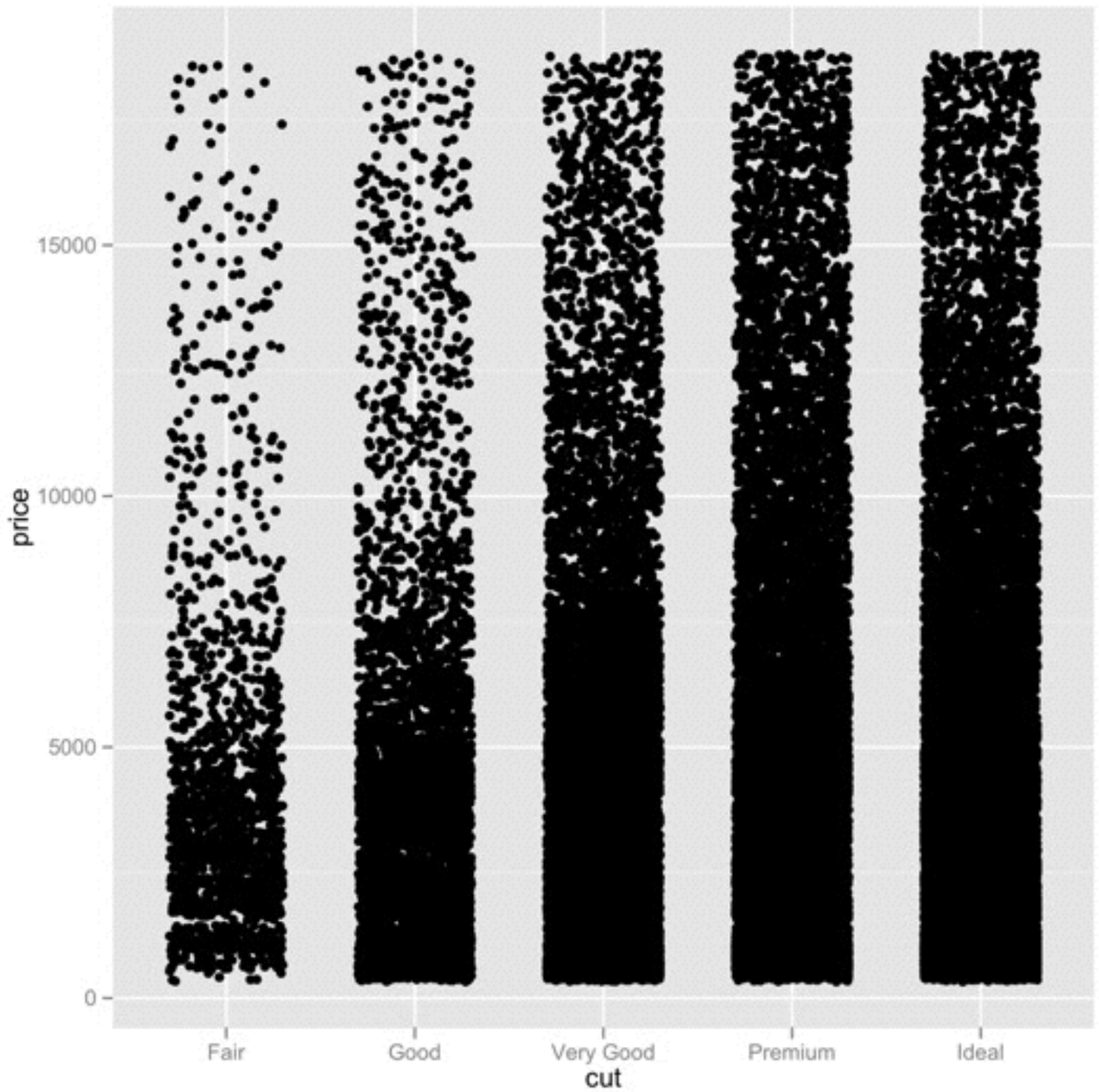
Any idea how to reduce overprinting of points and show distribution better?

plot ::= coord scale+ facet? layer+

la[illegible] data mapping stat geom position?

Is this better?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_jitter(width=0.3, height=0)
)
```

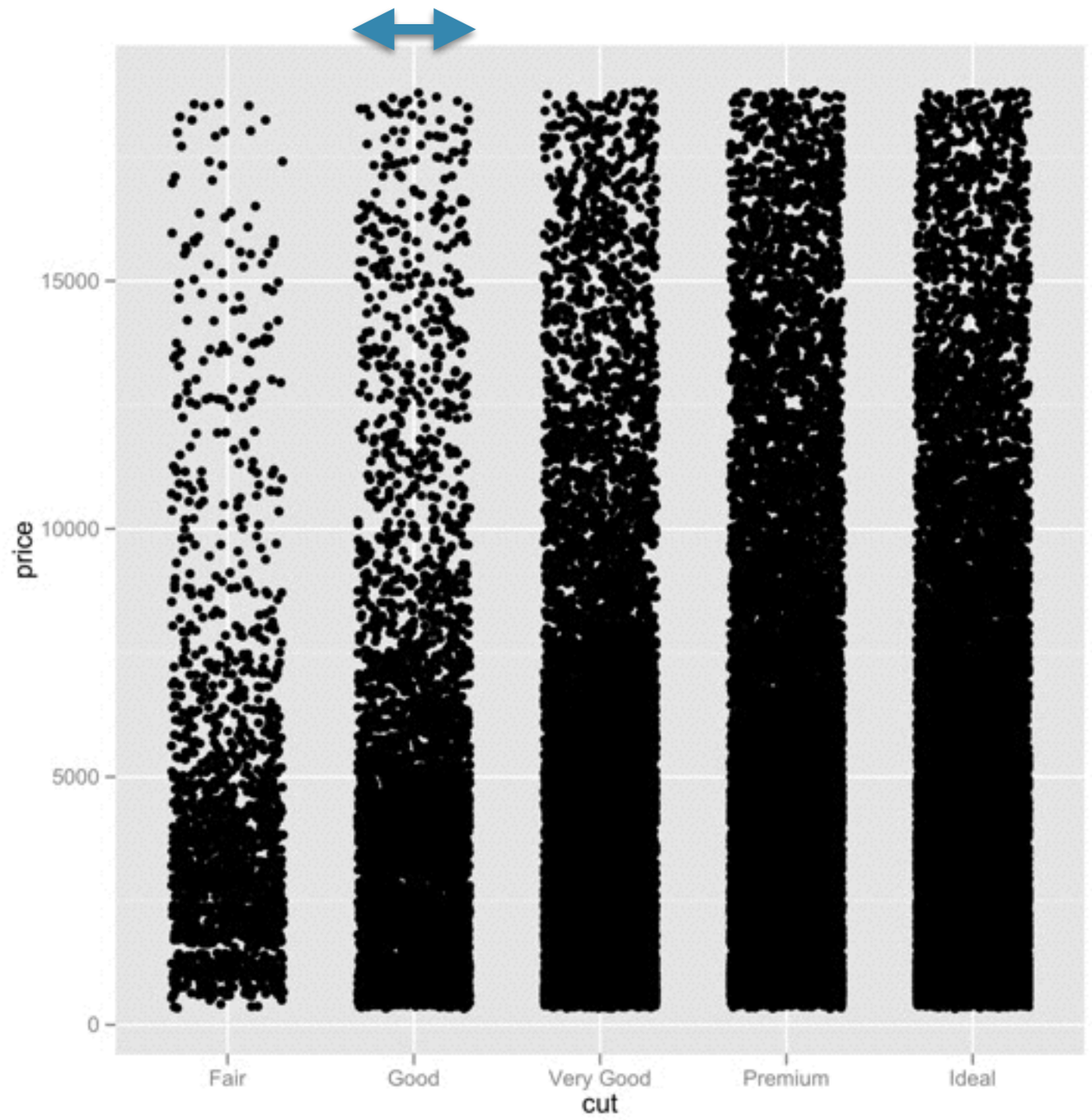


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=
  mapp
  stat=
  geom="point",    geom_params=list(),
  position=position_jitter(width=0.3, height=0)
)
```

add jitter
(+/- 0.3 horizontally)

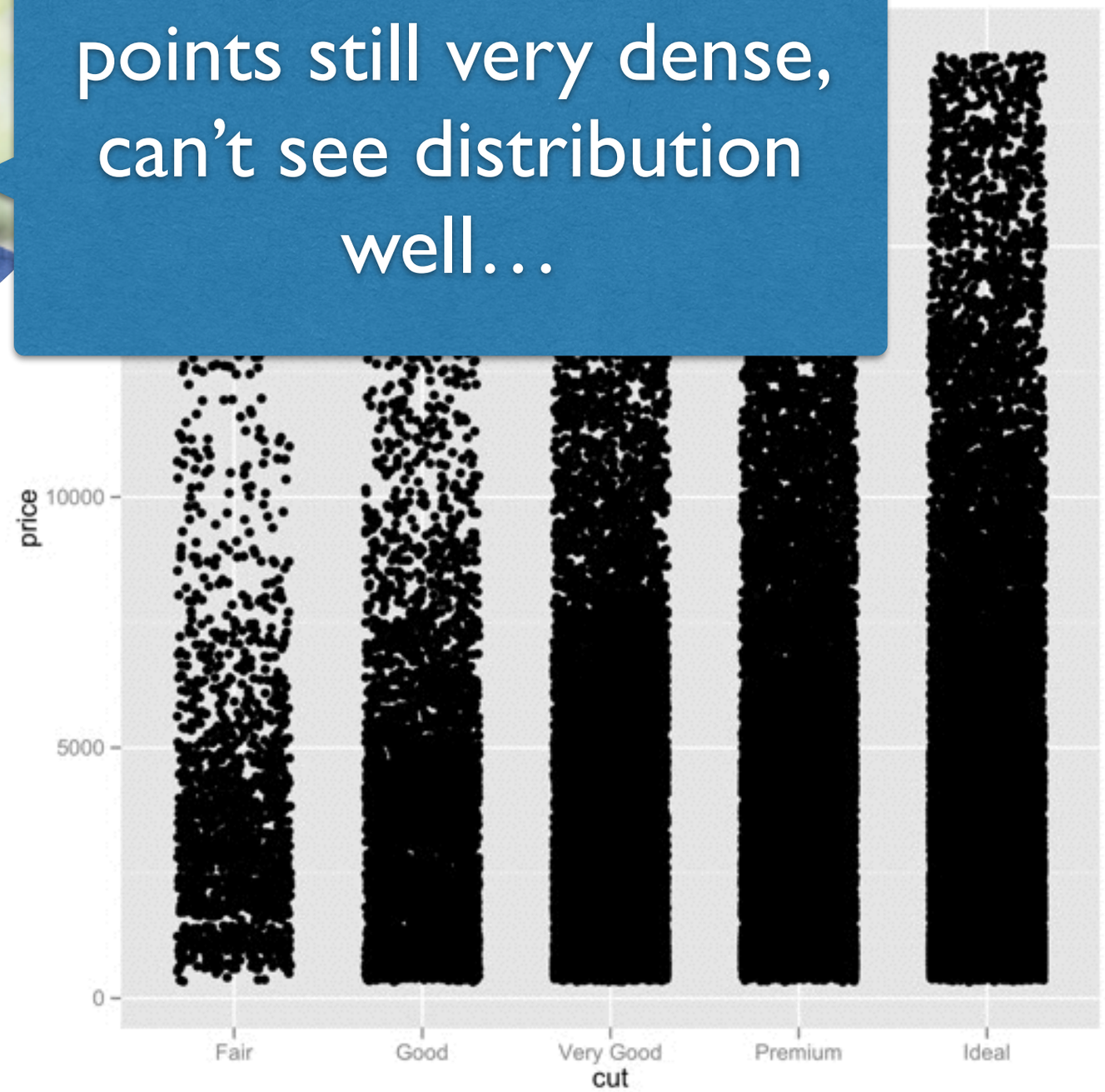


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

points still very dense, can't see distribution well…

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_jitter(width=0.3, height=0)
)
```



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Wasn't there some compact plot summarising distributions?

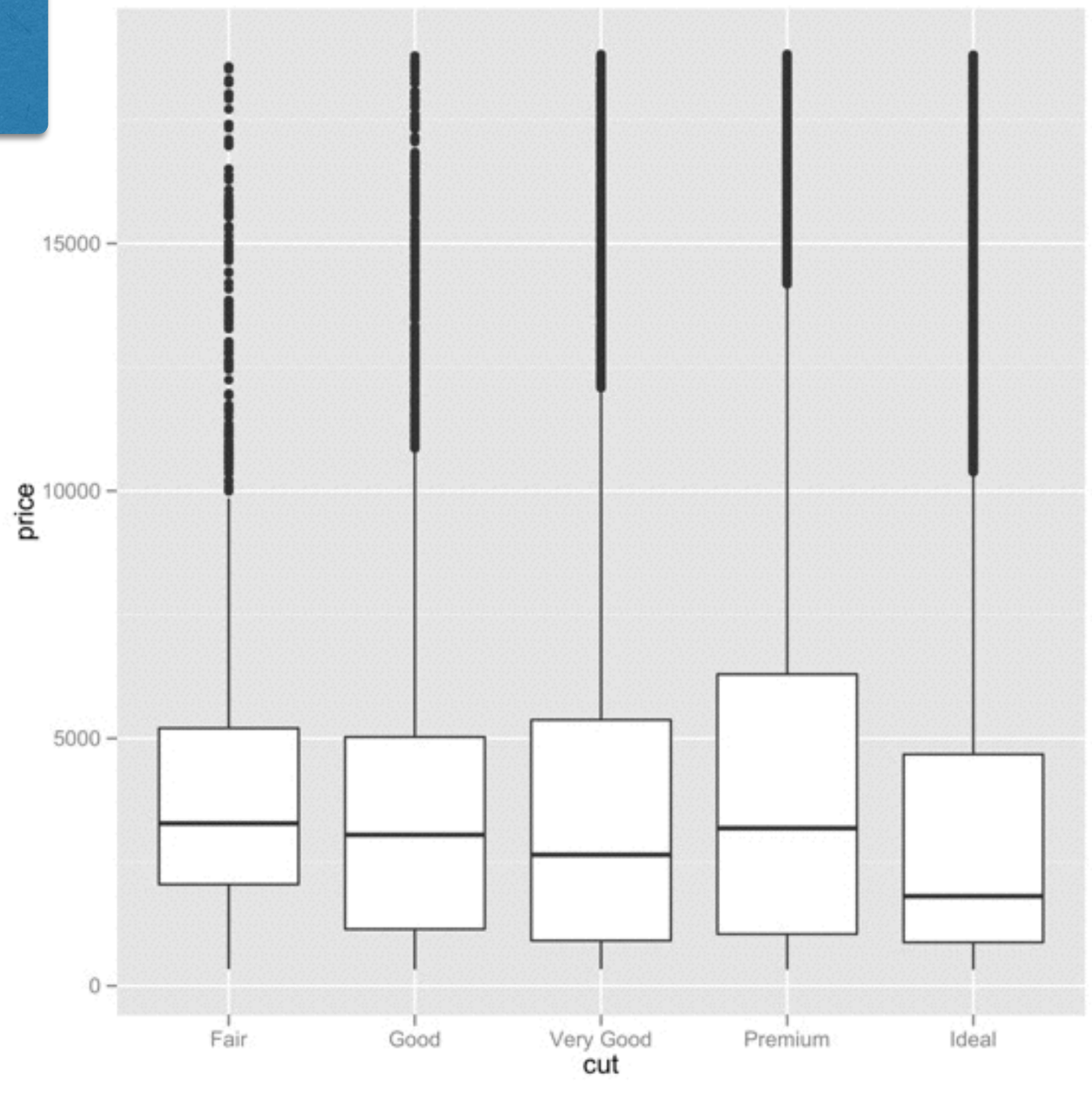
Starting with b…?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

How about this?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="boxplot", stat_params=list(),
  geom="boxplot", geom_params=list(),
  position=position_dodge()
)
```

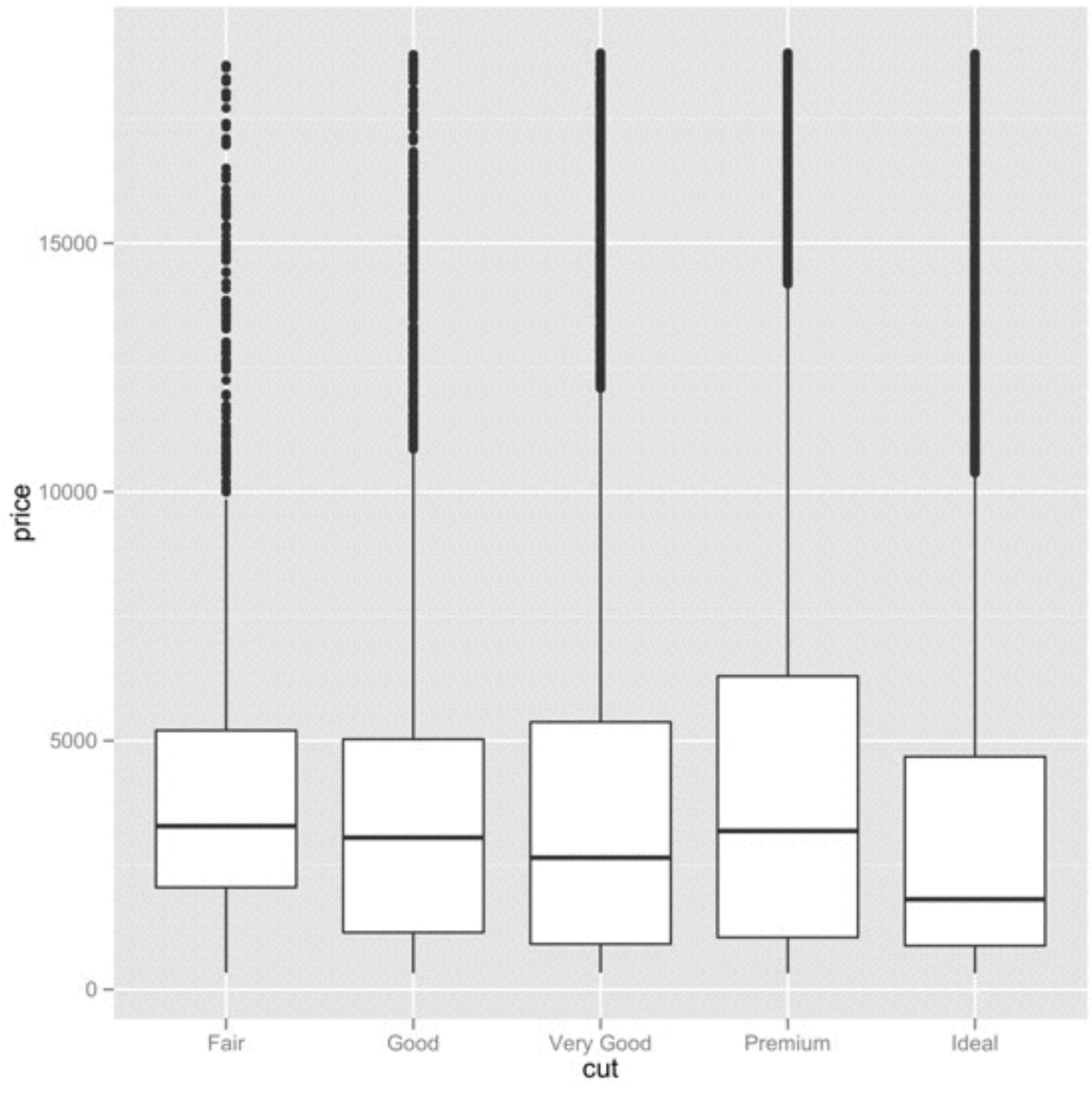


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
s
la
  (
  mapping=aes(x=cut, y=price),
  stat="boxplot", stat_params=list(),
  geom="boxplot", geom_params=list(),
  position=position_dodge()
)
```

summarize data for box plot

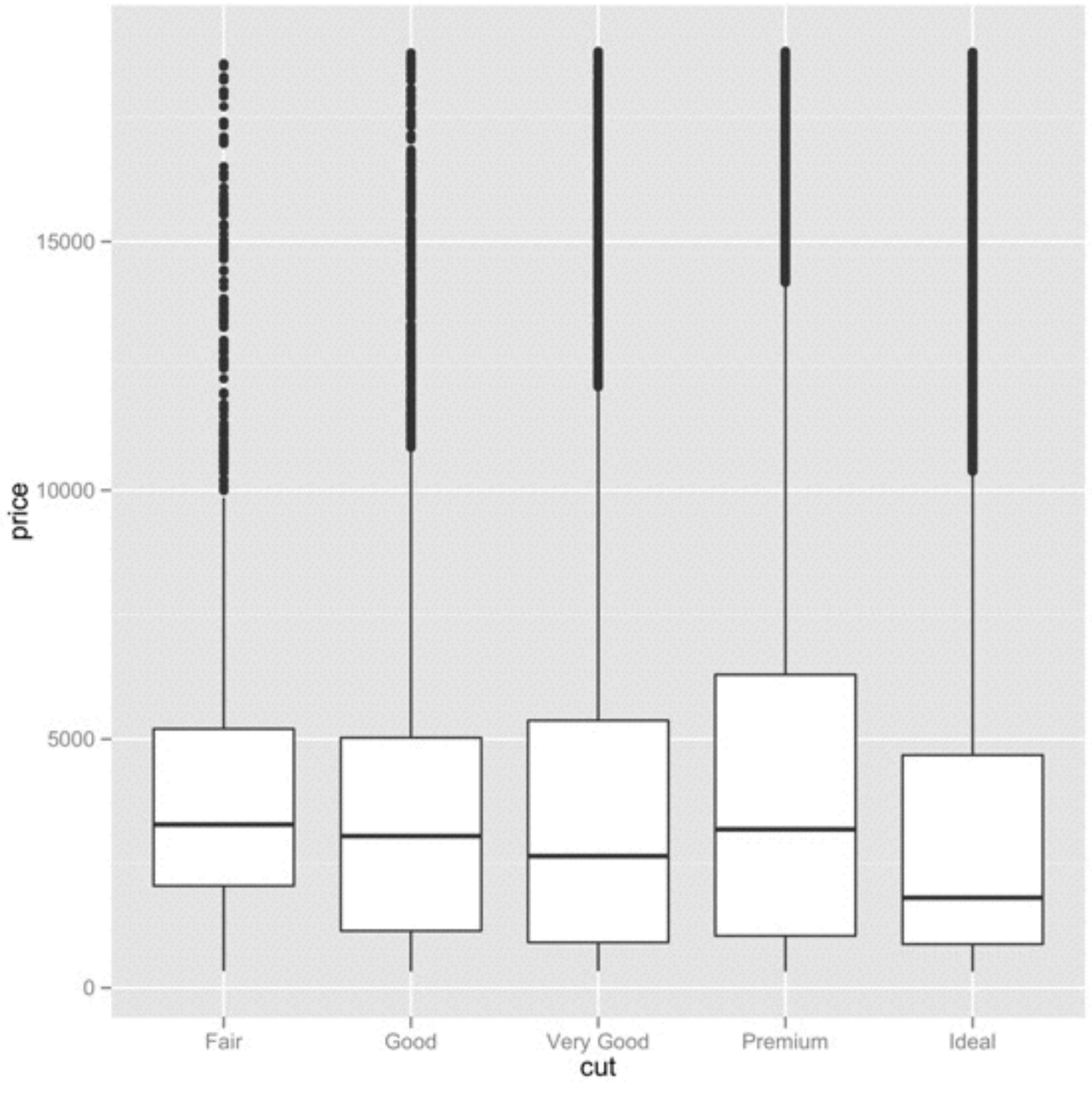


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
lay
  da
  m
  stat="boxplot", stat_params=list(),
  geom="boxplot", geom_params=list(),
  position=position_dodge()
)
```

and draw box plot

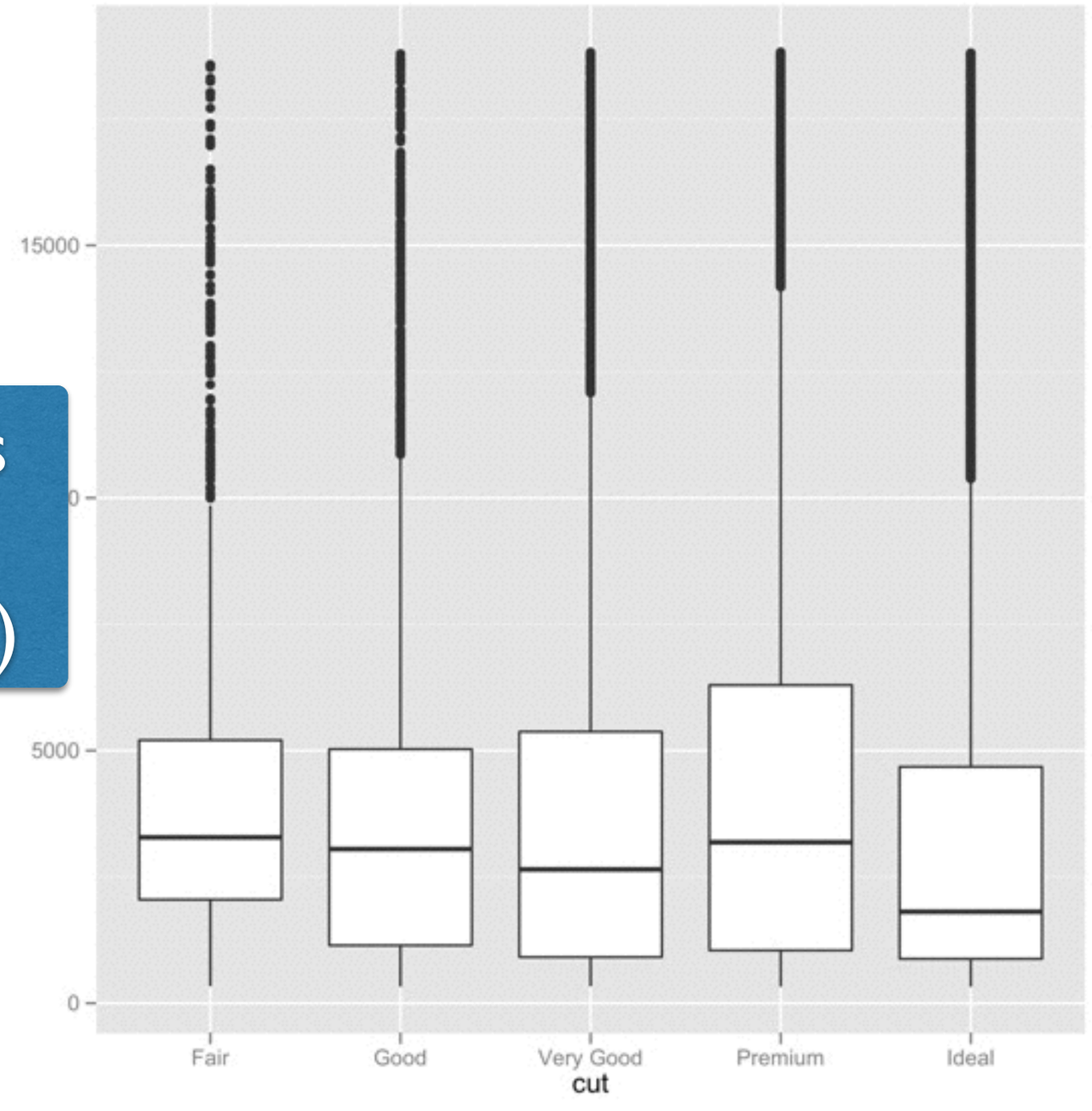


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_
layer(
  data=
  mapp
  stat=
  geom= boxplot", geom_params=list(),
  position=position_dodge()
)
```

dodge, if multiple boxes at same x (not really needed here)



plot ::= coord scale+ facet? layer+
layer ::= data mapping stat geom position?

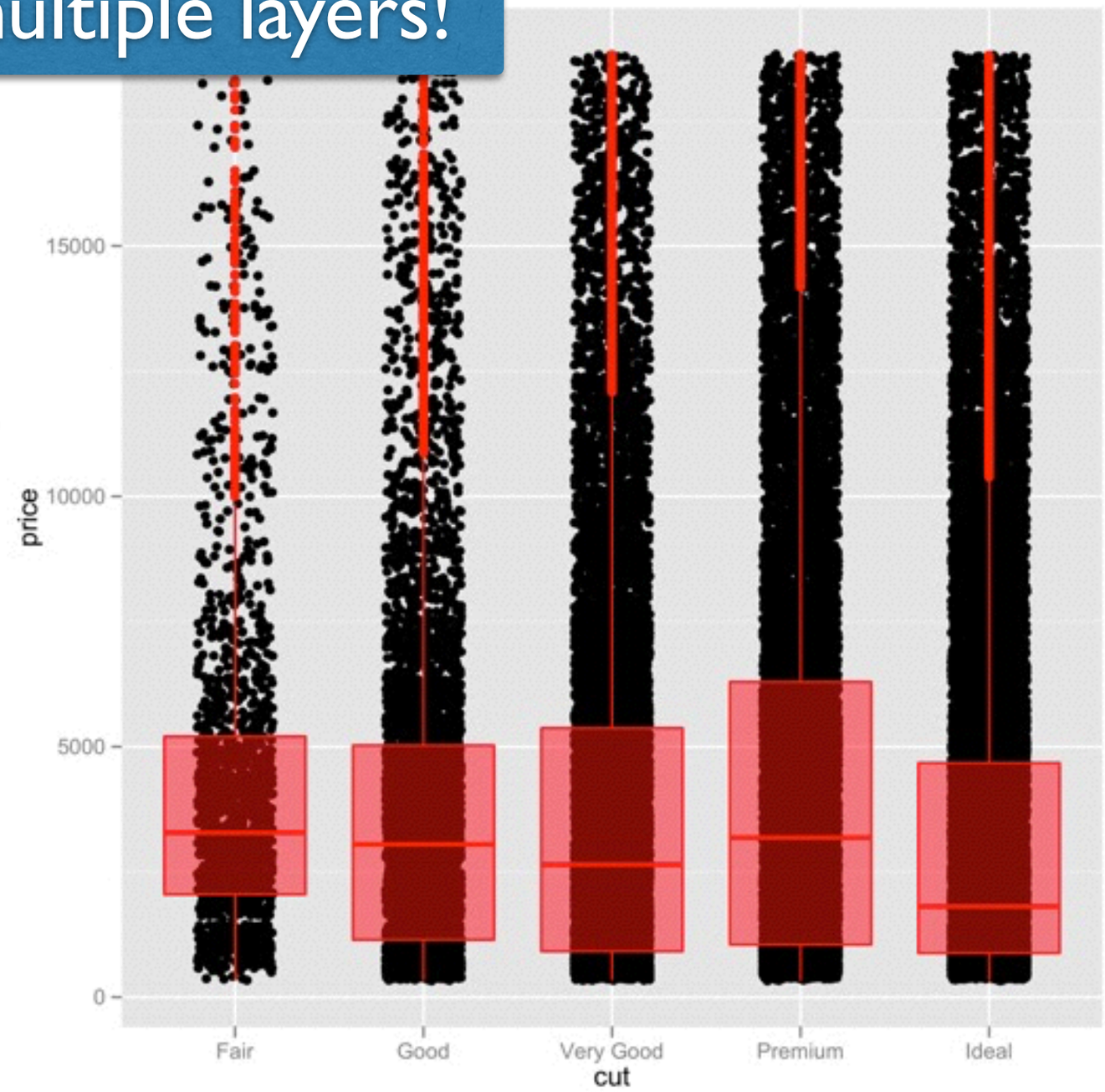
Could we combine box plot and jitter plot?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Sure! Just add multiple layers!

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_jitter(width=0.2, height=0)
) +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="boxplot", stat_params=list(),
  geom="boxplot",
  geom_params=
    list(color="red", fill="red", alpha=0.5),
  position=position_identity()
)
```

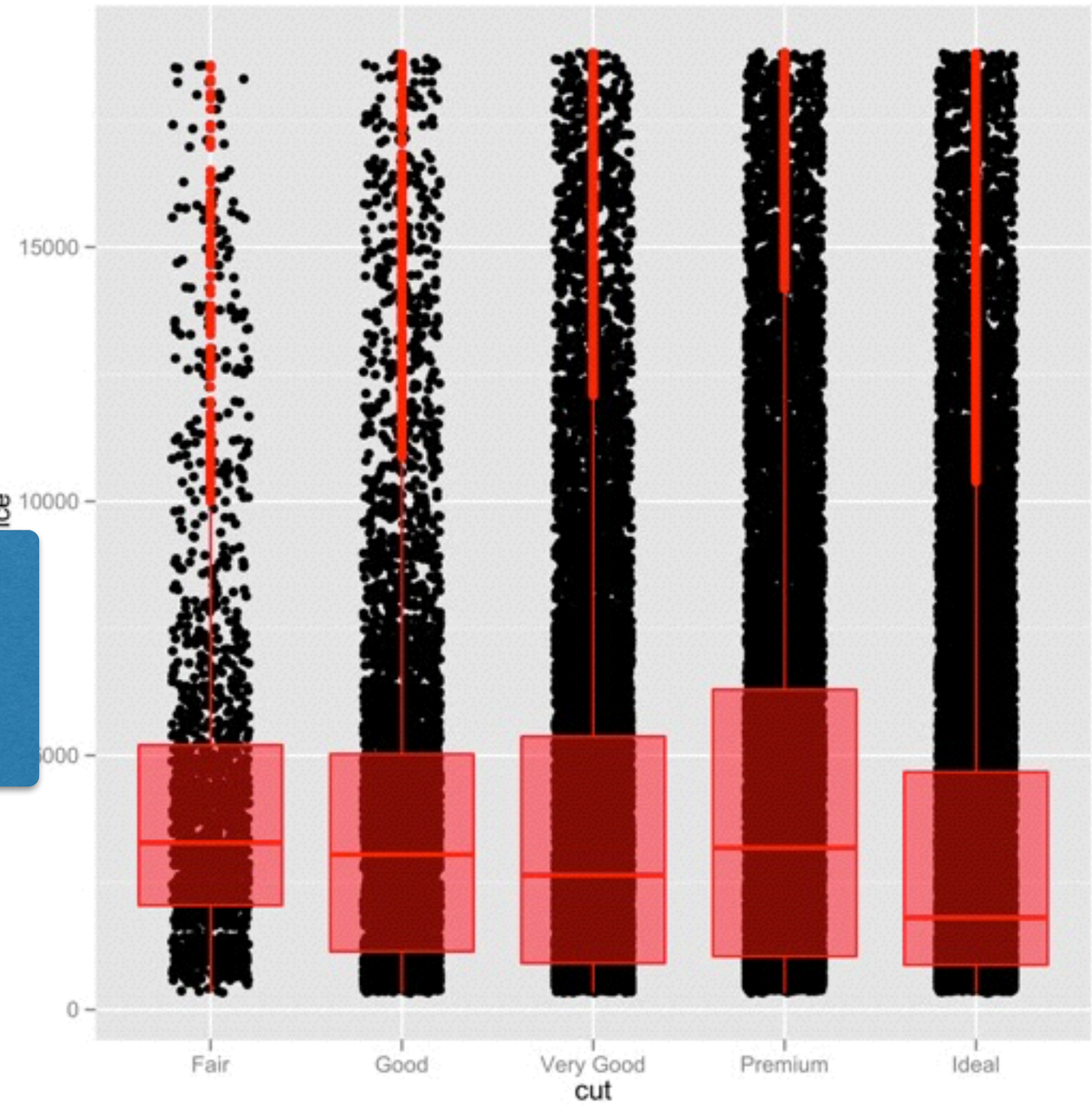


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=cut, y=price),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_jitter(width=0.2, height=0)
) +
layer(
  data=diam
  mapping=
  stat="boxp
  geom="boxplo",
  geom_params=
    list(color="red", fill="red", alpha=0.5),
  position=position_identity()
)
```

make box plots red, fill 50% transparent



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

I like color!

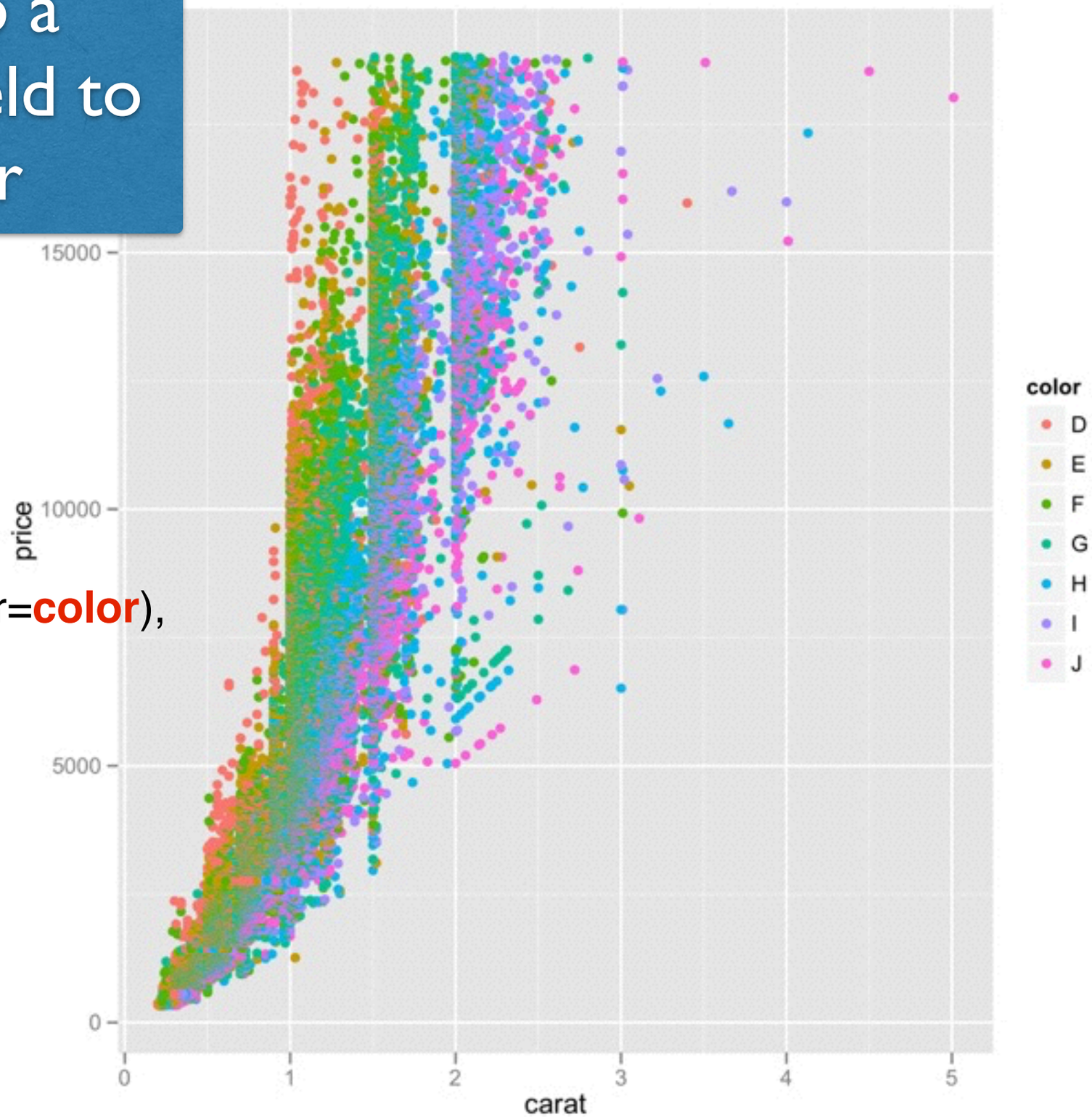
Can we use different colors depending on the data?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Sure! Map a data.table field to the color

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
scale_color_hue() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

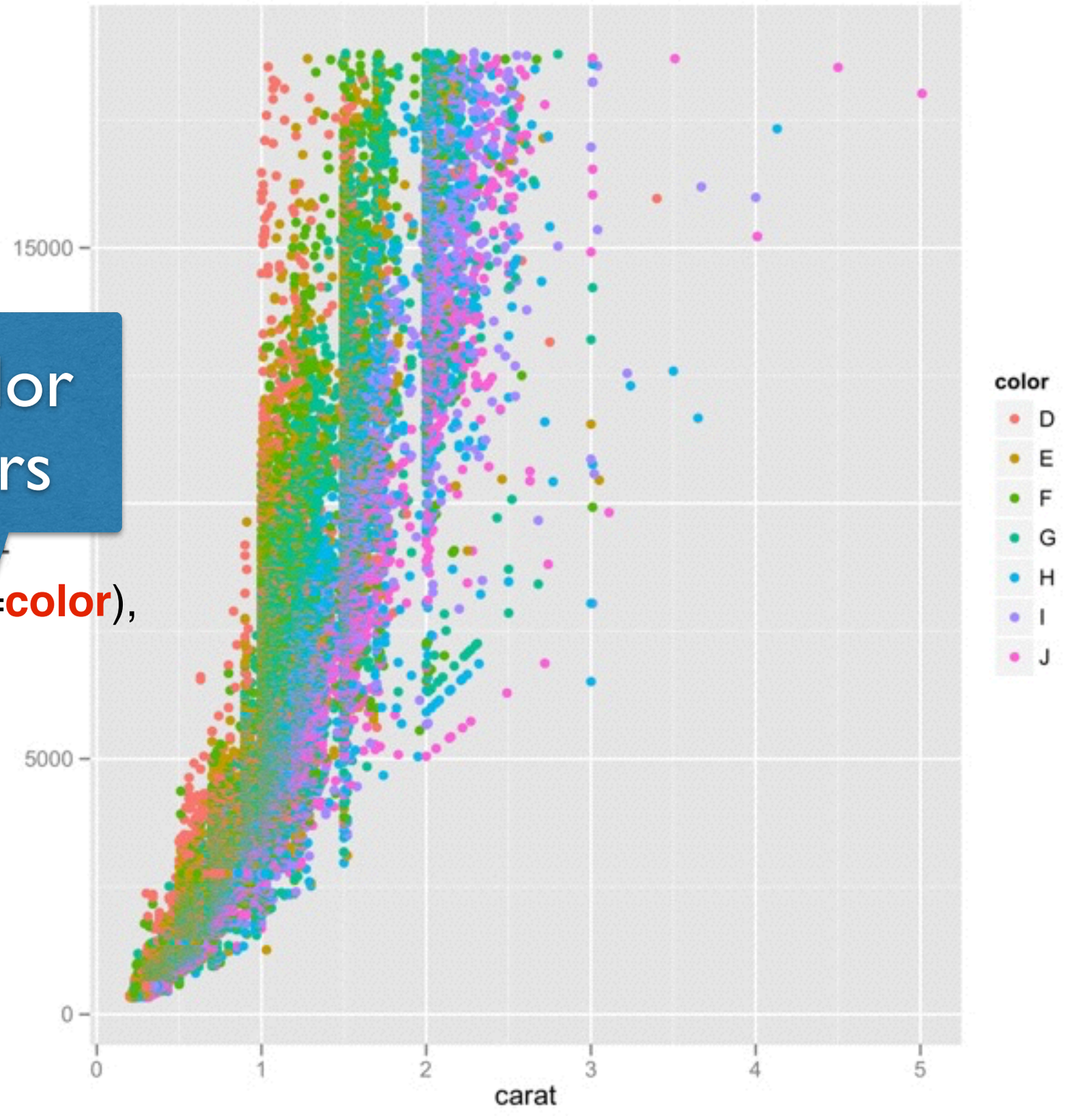


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
sca
sca
sca
laye
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

Map the diamonds$color field to aesthetic colors

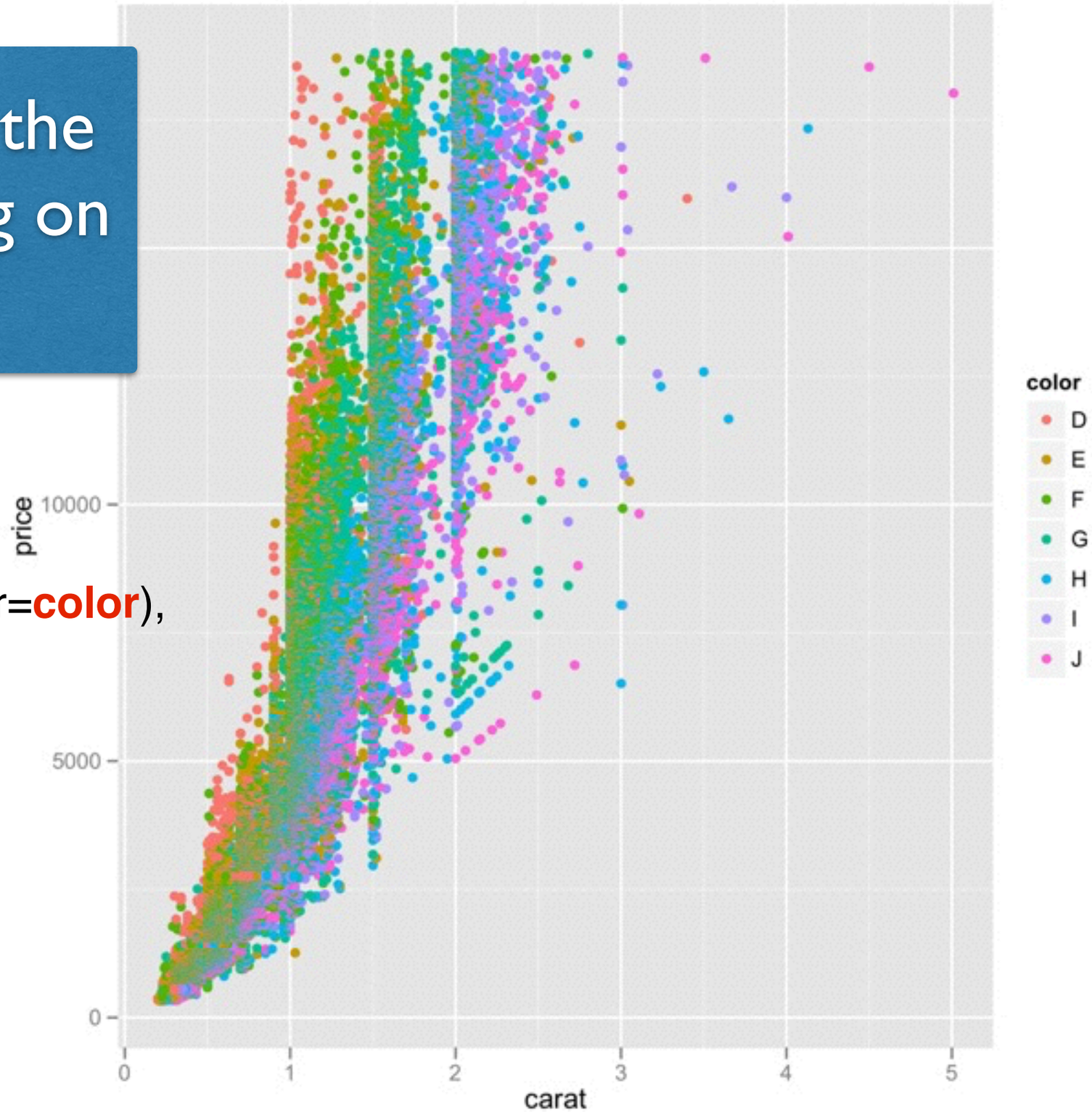


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Set the color scale so the **hue** changes depending on the data value

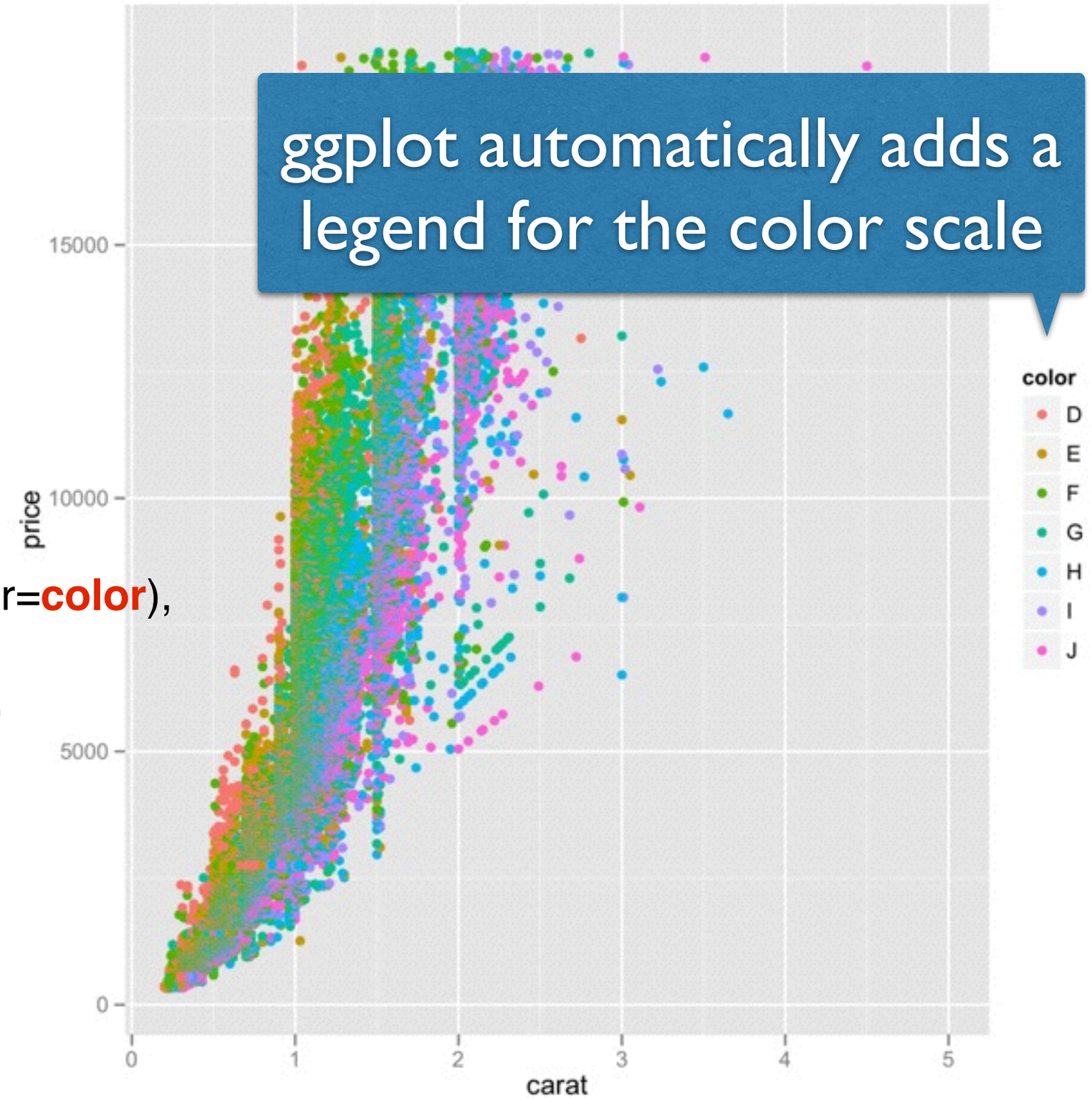
```
ggp
co
sca
scale_y_continuous() +
scale_color_hue() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
scale_color_hue() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

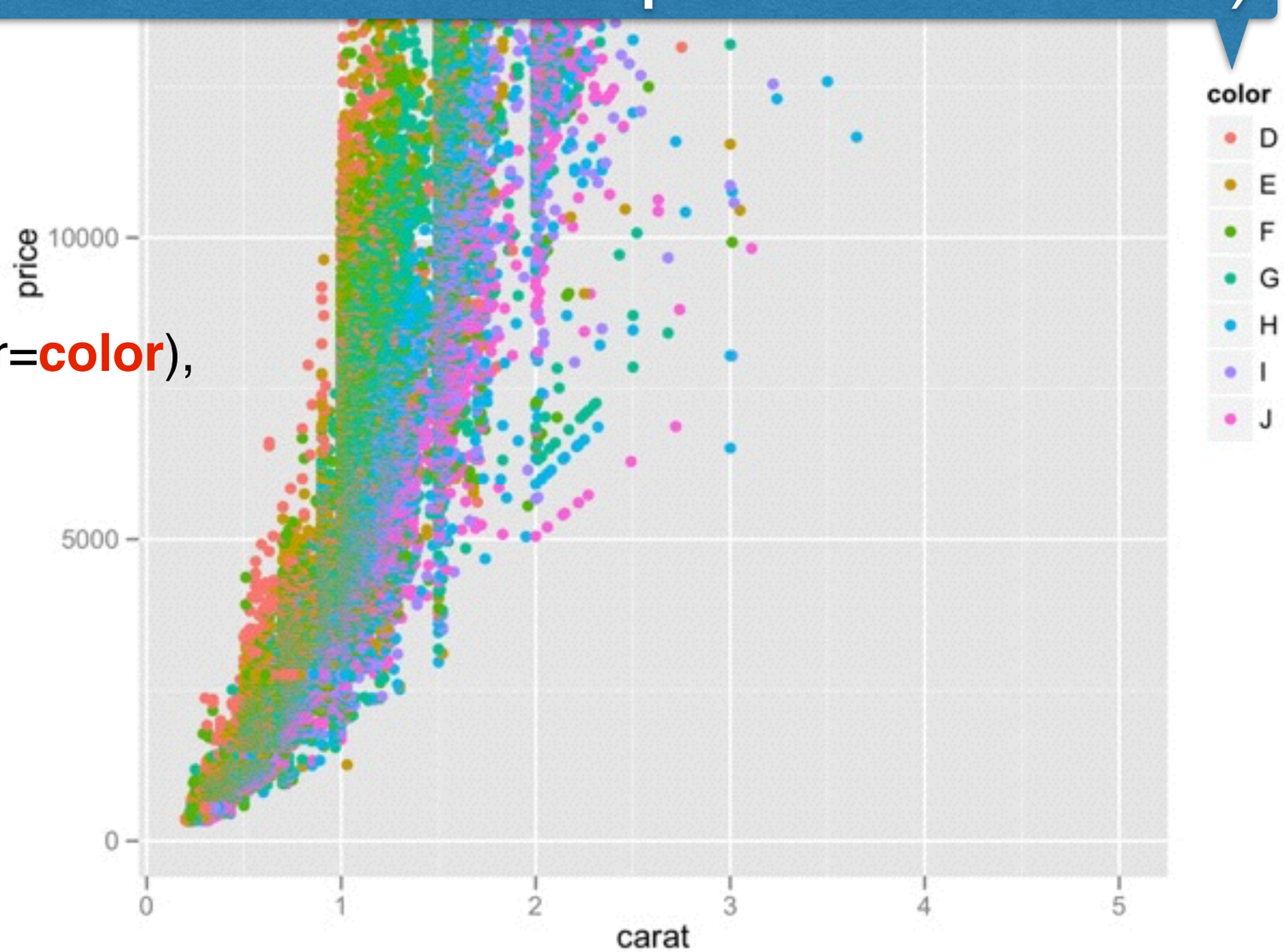


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

The levels of the diamonds$color factor are:
D, E, F, G, H, I, J
(this is what diamond experts call "color")

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
scale_color_hue() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```

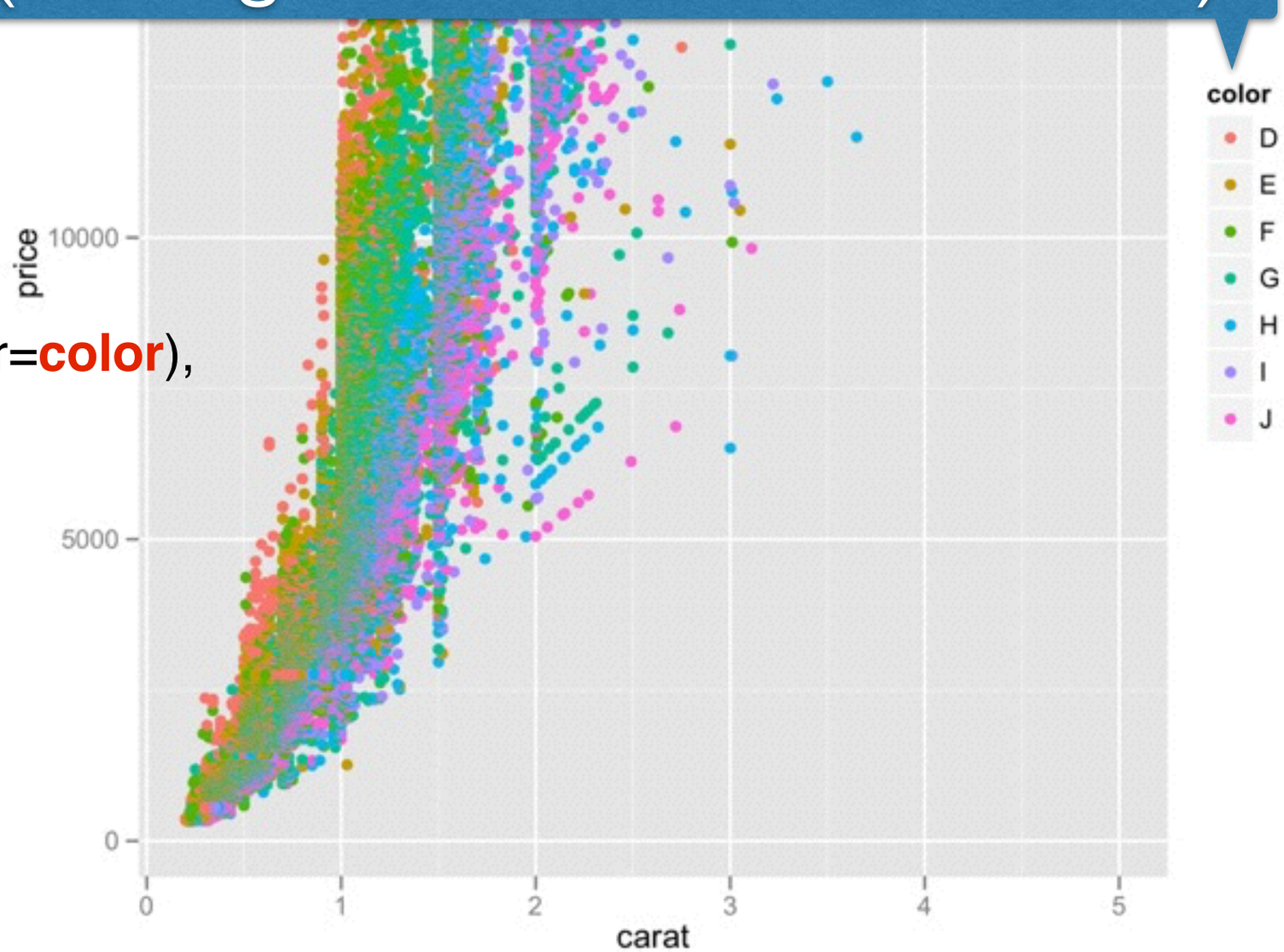


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Using the hue color scale,
ggplot automatically maps each level
to a distinct hue
(starting with red for the first level)

```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
scale_color_hue() +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Still a ton of data points.

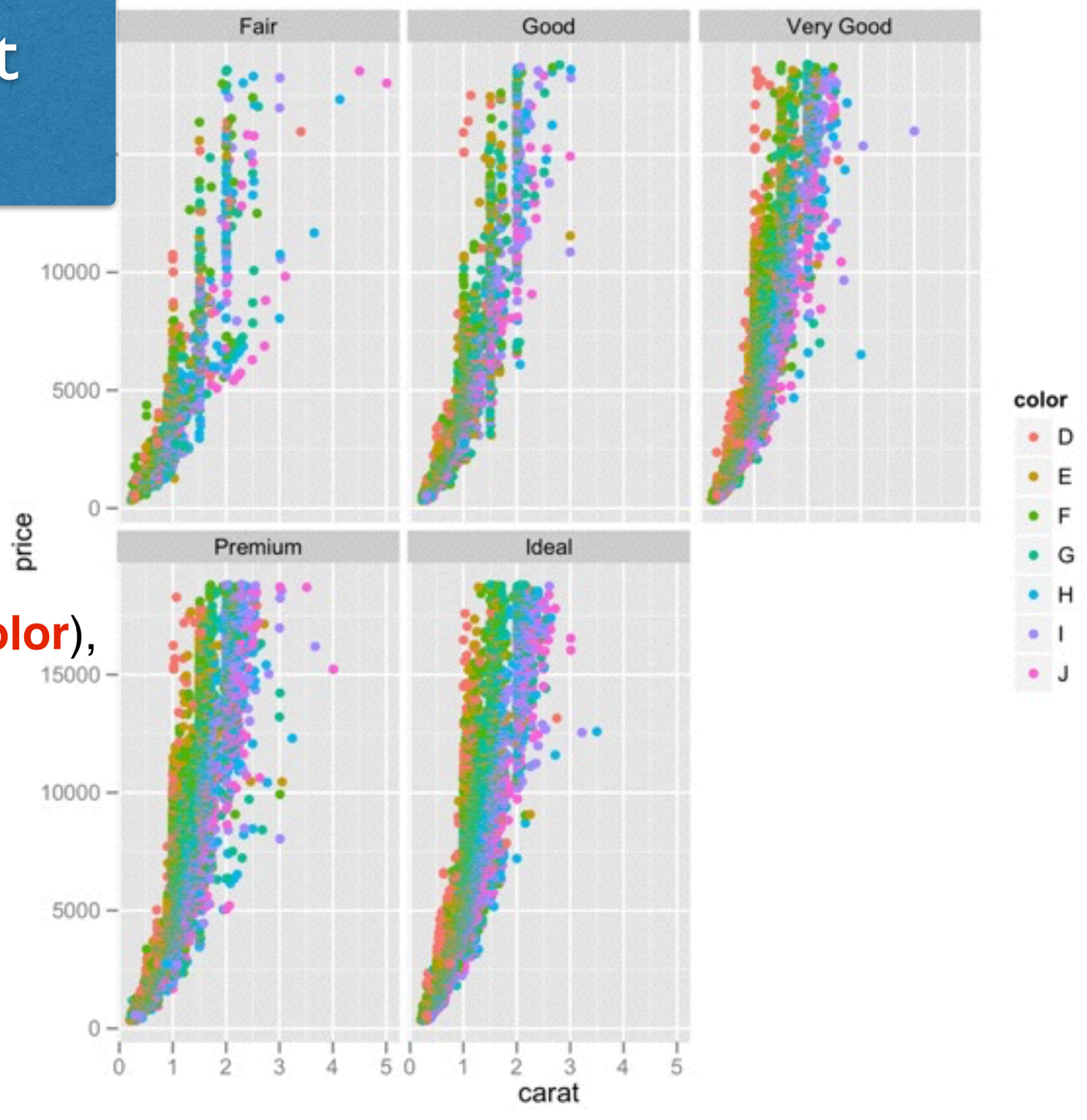
Can we break the data down somehow?

plot ::= coord scale+ facet? layer+

l = data mapping stat geom position?

Sure! Break into a separate facet for each cut.

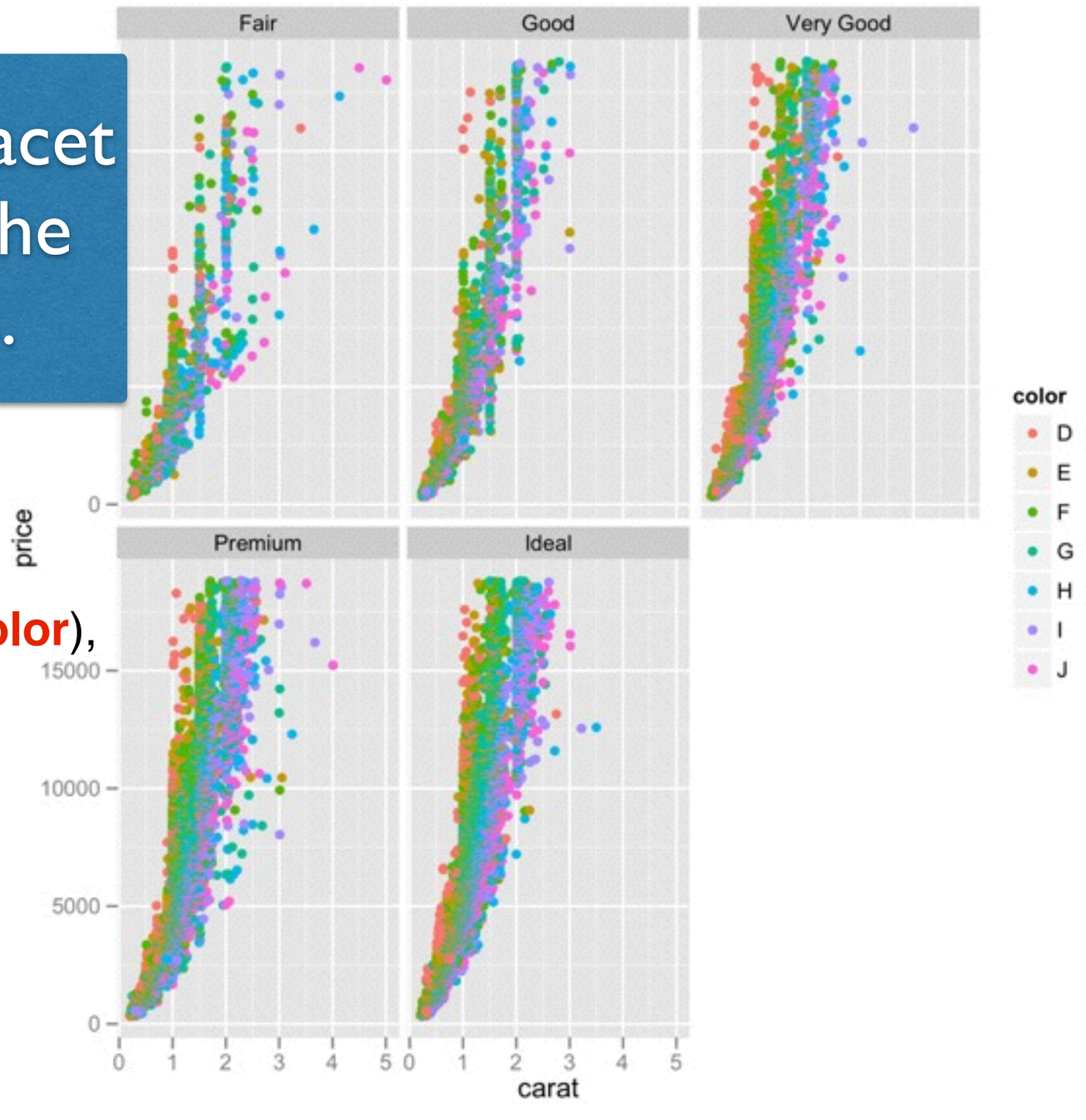
ggplot() +
coord_cartesian() +
scale_**x**_continuous() +
scale_**y**_continuous() +
scale_**color**_hue() +
facet_wrap(~**cut**) +
layer(
  data=**diamonds**,
  mapping=aes(x=**carat**, y=**price**, color=**color**),
  stat="identity", stat_params=list(),
  geom="point", geom_params=list(),
  position=position_identity()
)



plot ::= coord scale+ facet? layer+
layer ::= data mapping stat geom position?

This tell ggplot to create a facet for each cut, and to wrap the facets into rows/columns.

```
scale_color_hue() +
facet_wrap(~cut) +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
)
```



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

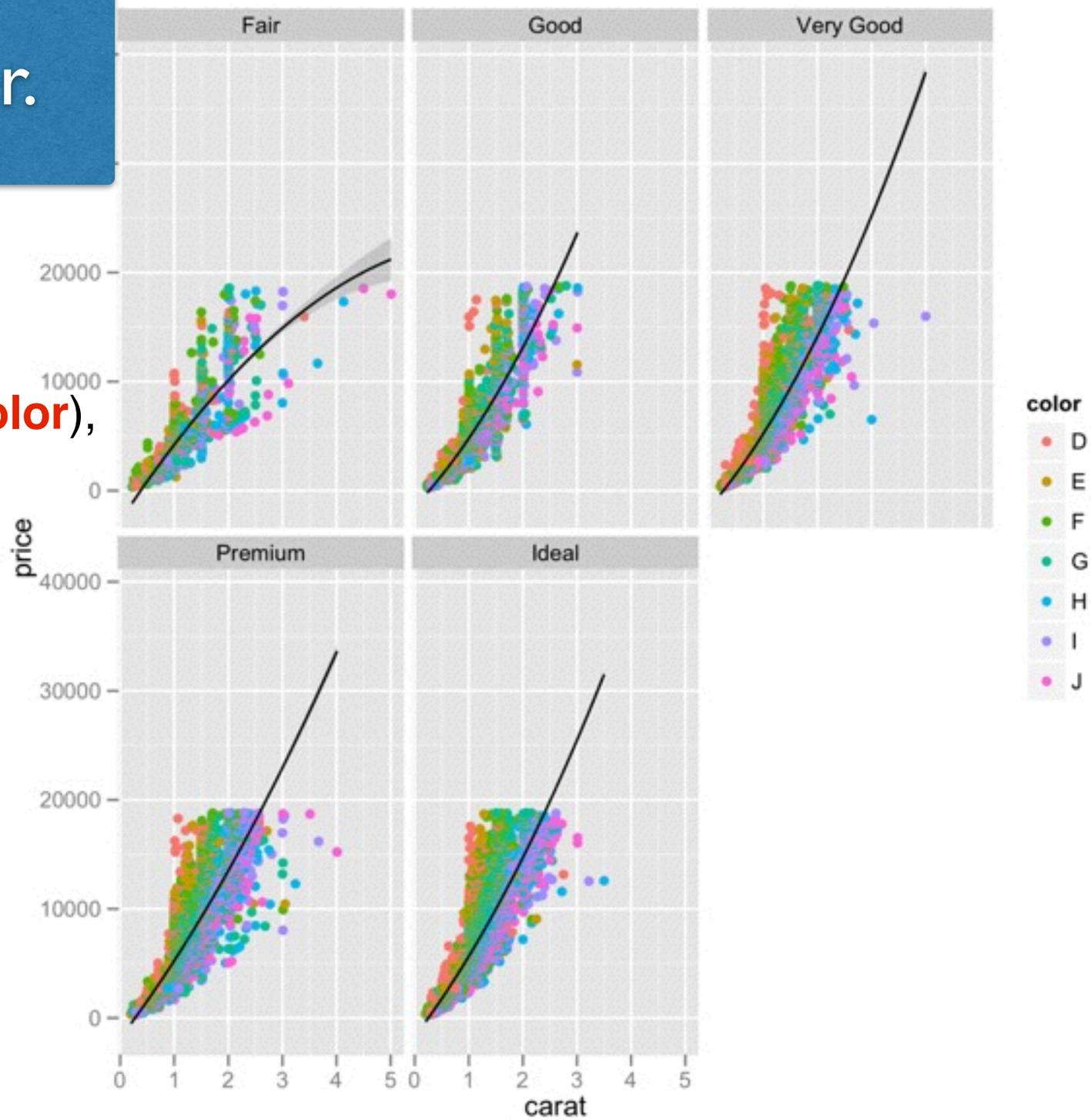
Can we add a fitted curve to the plot?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Sure!
Just add a layer.

```
g
d
s
sca                    )
scale_color_hue() +
facet_wrap(~cut) +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom="point",    geom_params=list(),
  position=position_identity()
) +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price),
  stat="smooth", stat_params=list(),
  geom="smooth", geom_params=list(),
  position=position_identity()
)
```

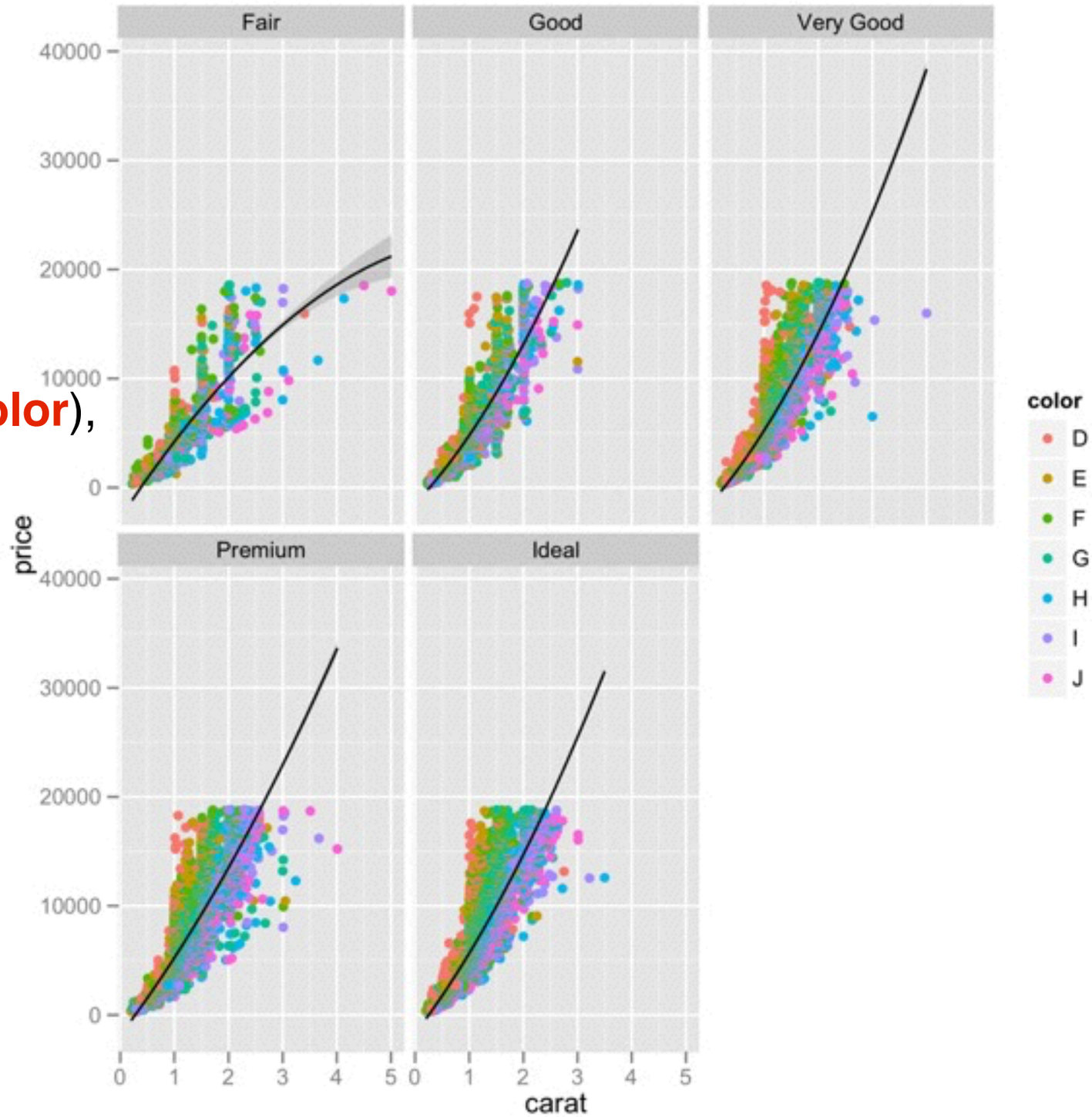


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

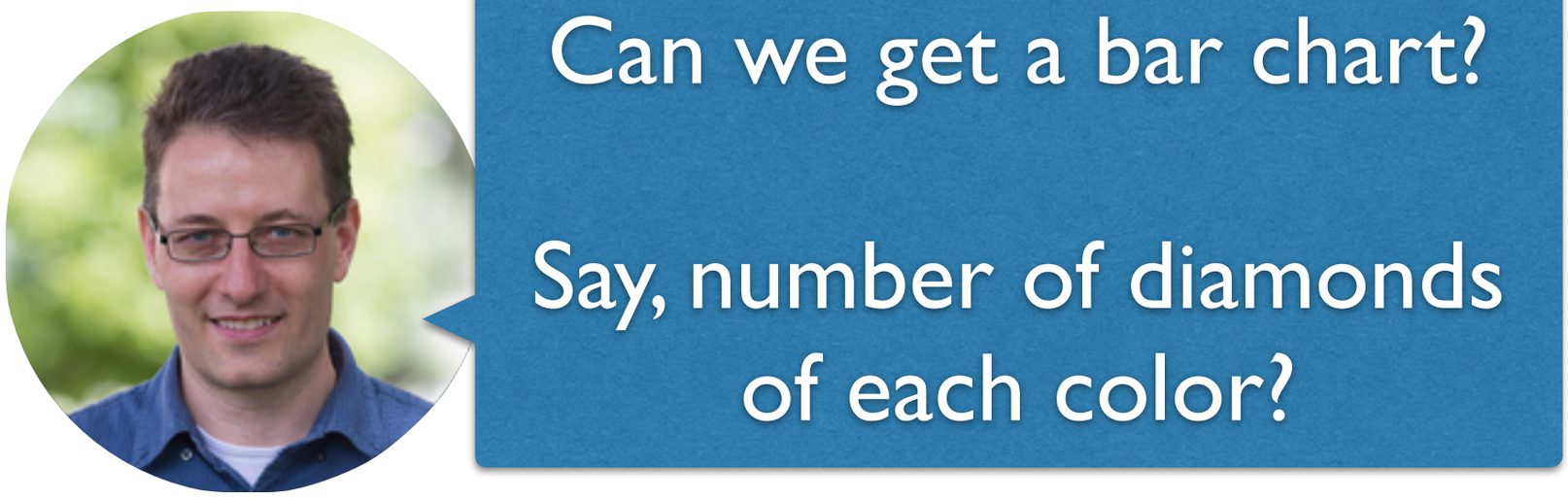
```
ggplot() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous() +
scale_color_hue() +
facet_wrap(~cut) +
layer(
  data=diamonds,
  mapping=aes(x=carat, y=price, color=color),
  stat="identity", stat_params=list(),
  geom=
  positio
) +
layer(
  data=d
  mapping=aes(x=carat, y=price),
  stat="smooth", stat_params=list(),
  geom="smooth", geom_params=list(),
  position=position_identity()
)
```

Set statistic and geom to “smooth”



plot ::= coord scale+ facet? layer+

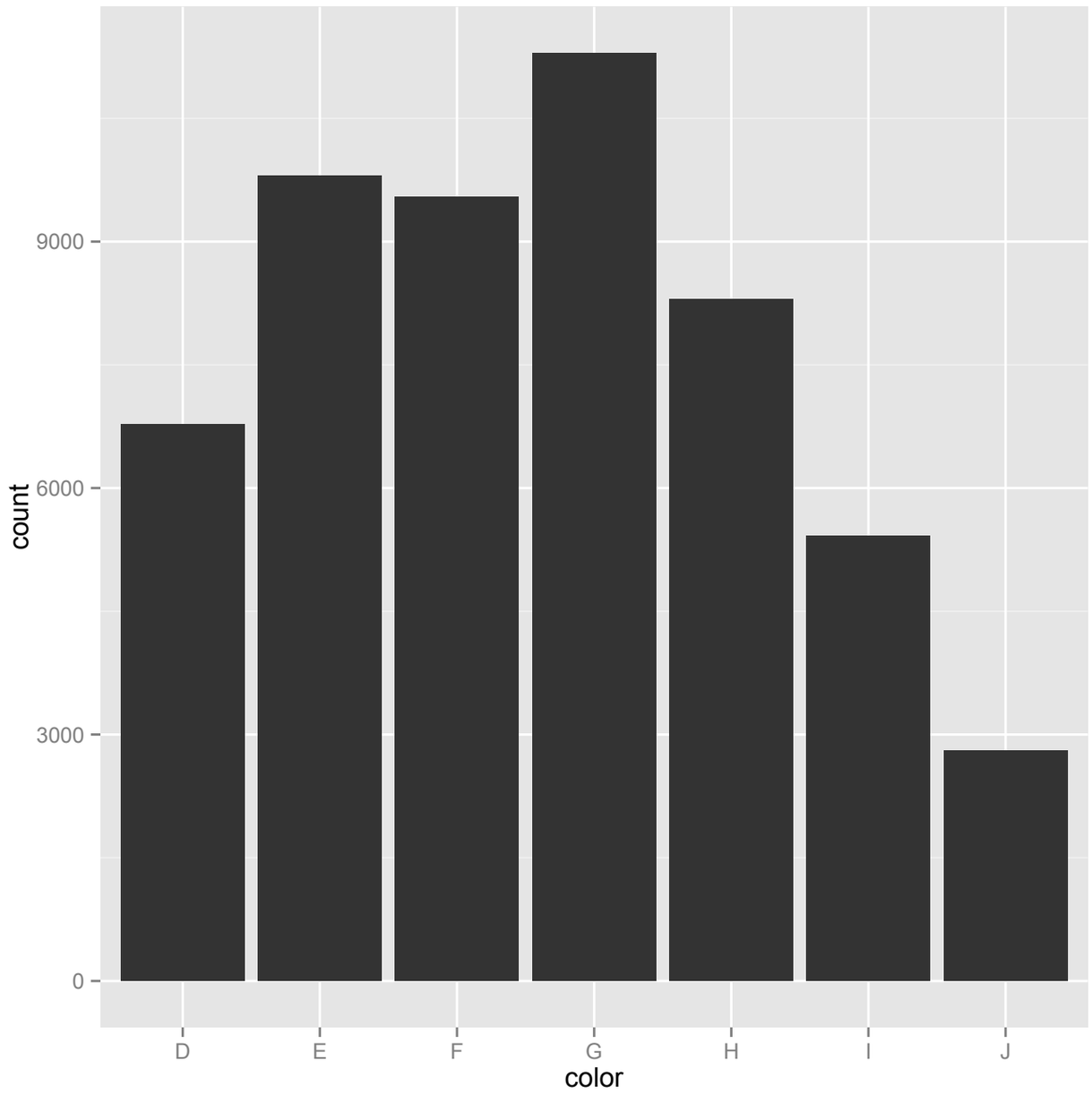
layer ::= data mapping stat geom position?



plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?


```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
layer(
  data=diamonds,
  mapping=aes(x=color),
  stat="bin", stat_params=list(),
  geom="bar", geom_params=list(),
  position=position_identity()
)
```

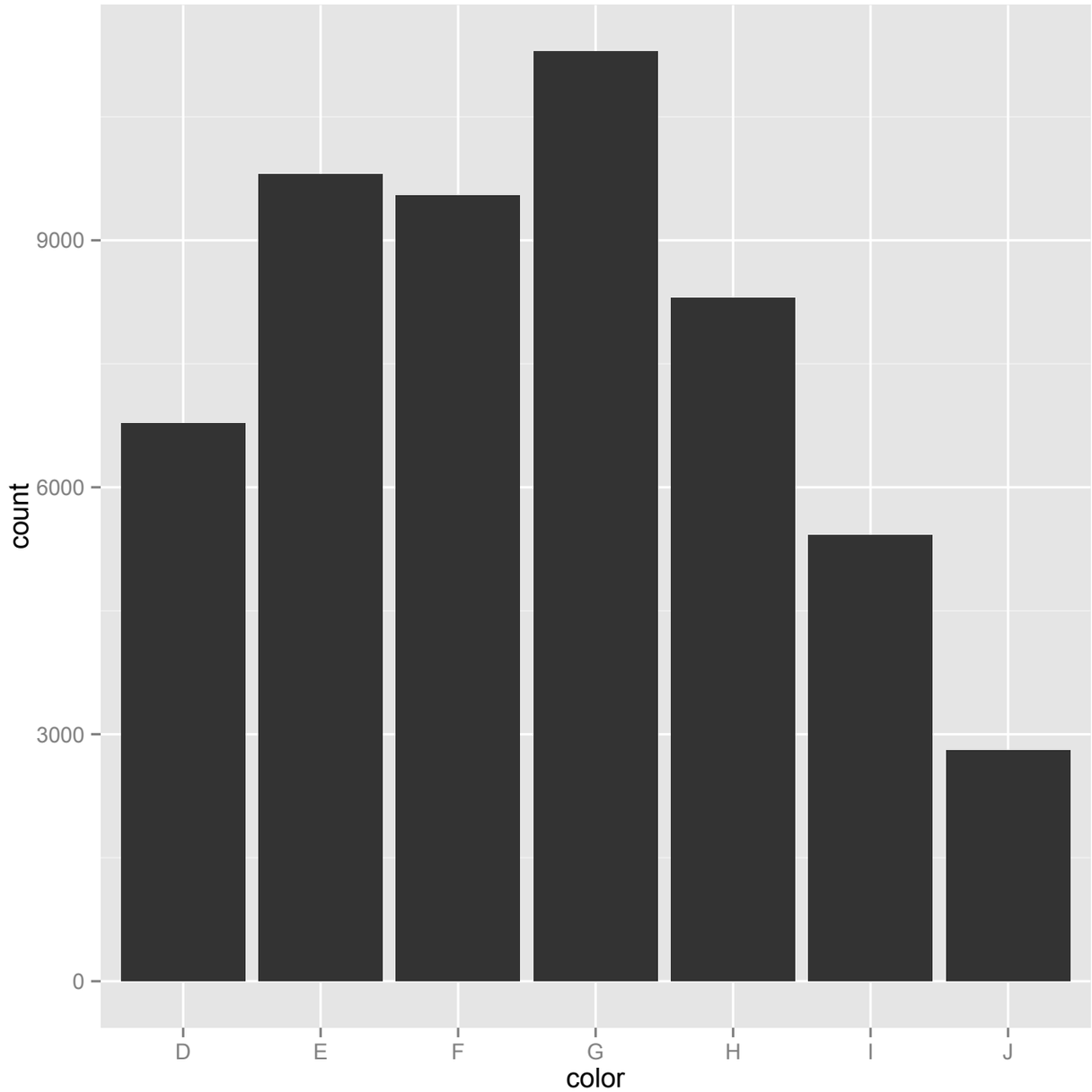


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
sc
sc
lay
  d
  mapping=aes(x=color),
  stat="bin", stat_params=list(),
  geom="bar", geom_params=list(),
  position=position_identity()
)
```

Bin the observations

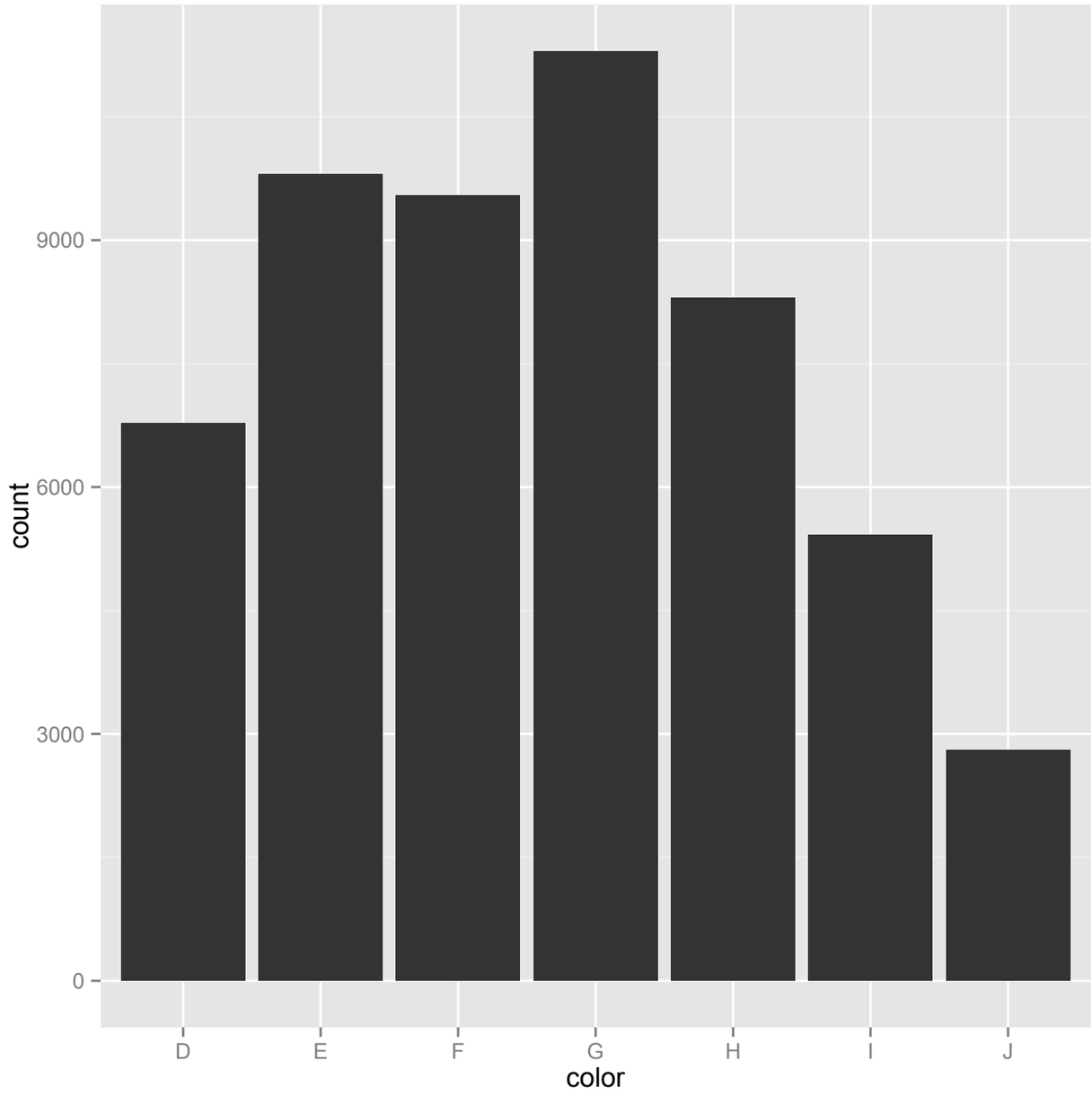


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scal
laye
  dat
  ma
  stat="bin", stat_params=list(),
  geom="bar", geom_params=list(),
  position=position_identity()
)
```

Draw a bar for each bin.



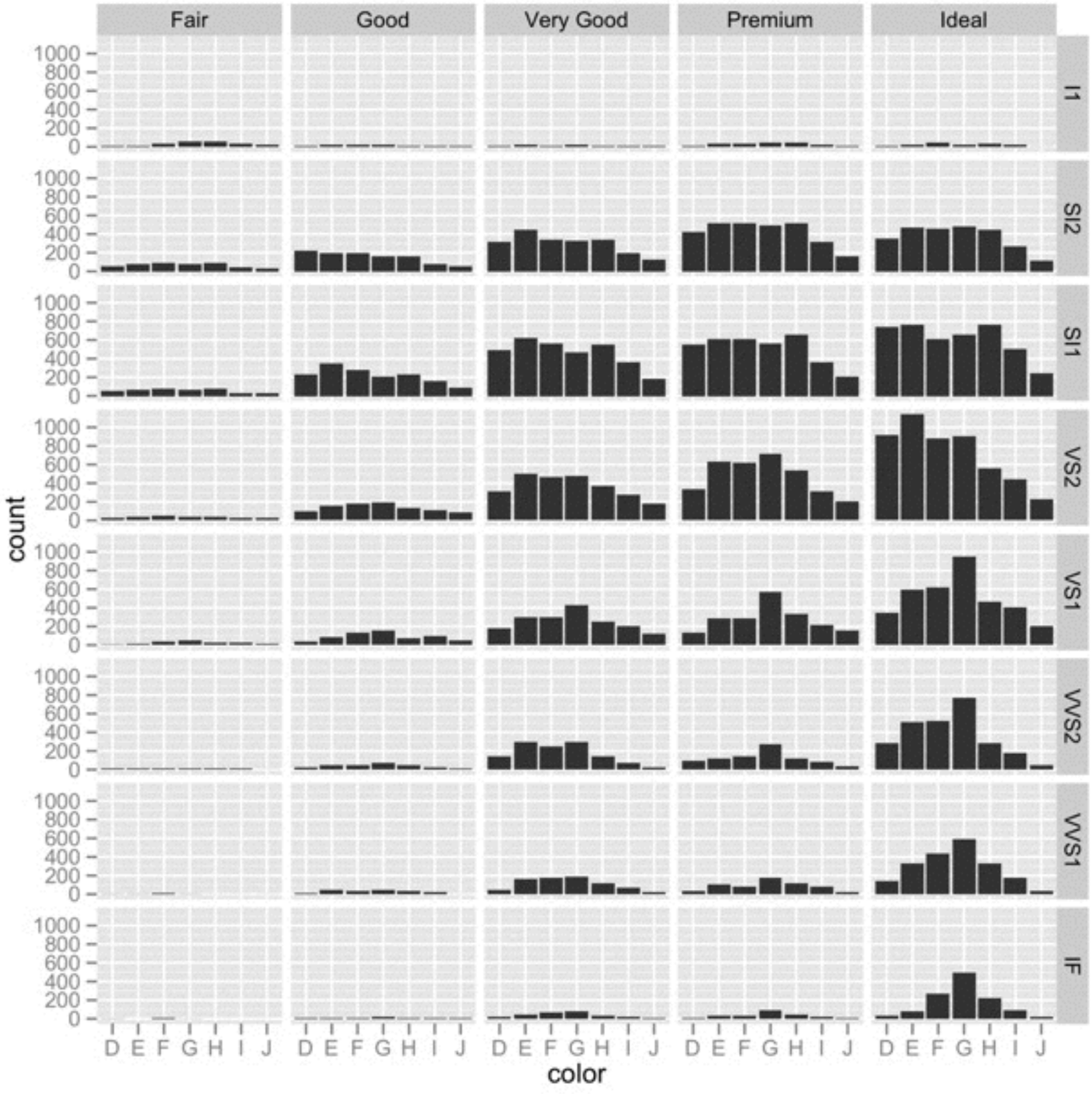
plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?


```
ggplot() +
coord_cartesian() +
scale_x_discrete() +
scale_y_continuous() +
facet_grid(clarity~cut) +
layer(
  data=diamonds,
  mapping=aes(x=color),
  stat="bin", stat_params=list(),
  geom="bar", geom_params=list(),
  position=position_identity()
)
```

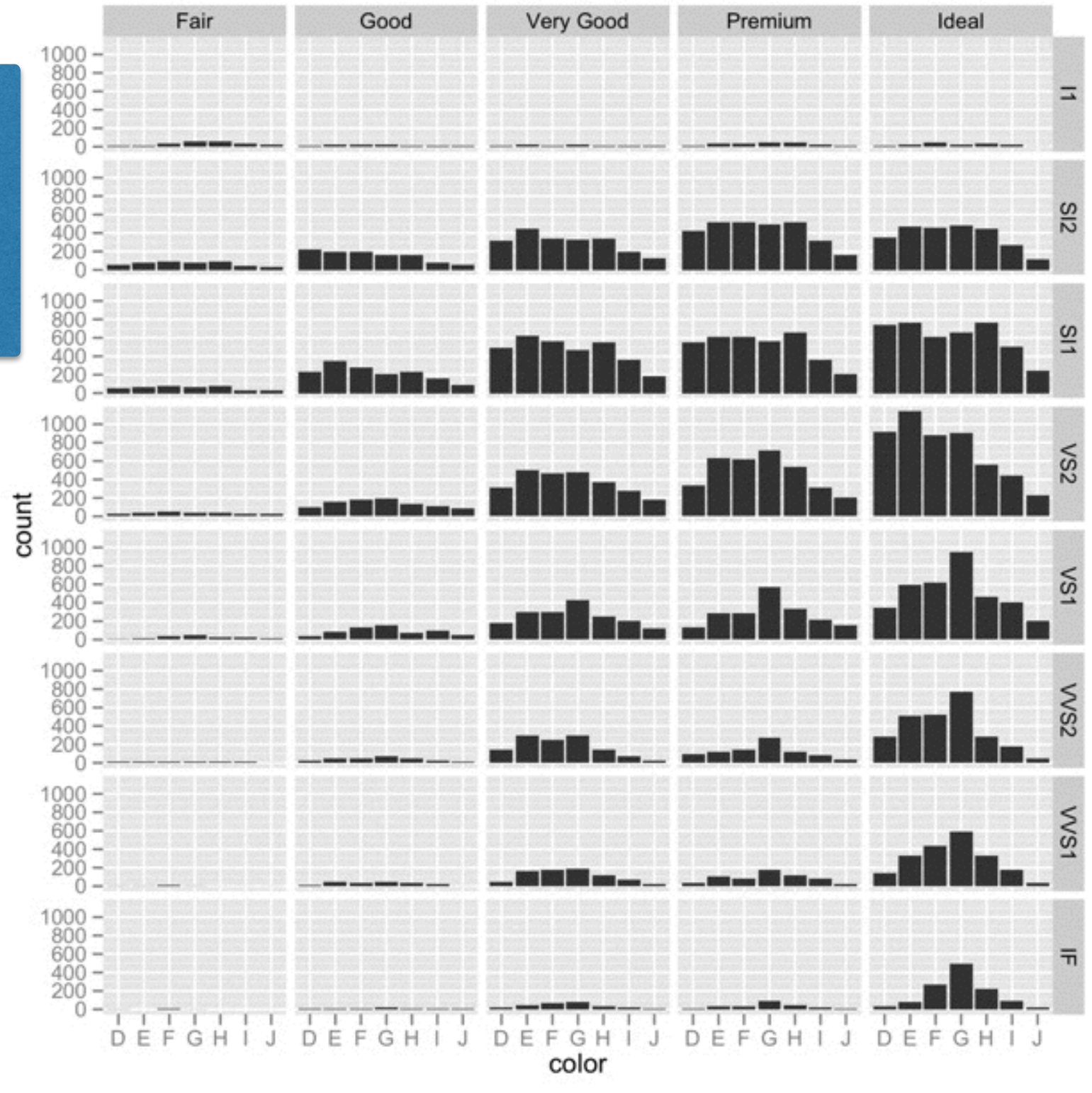


plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Create a grid of facets
rows: clarity
columns: cut

```
gg
co
sc
scale_y_continuous() +
facet_grid(clarity~cut) +
layer(
  data=diamonds,
  mapping=aes(x=color),
  stat="bin", stat_params=list(),
  geom="bar", geom_params=list(),
  position=position_identity()
)
```



Faceting's Panels
facet_...(...)

labels
breaks
minor_breaks
name
limits

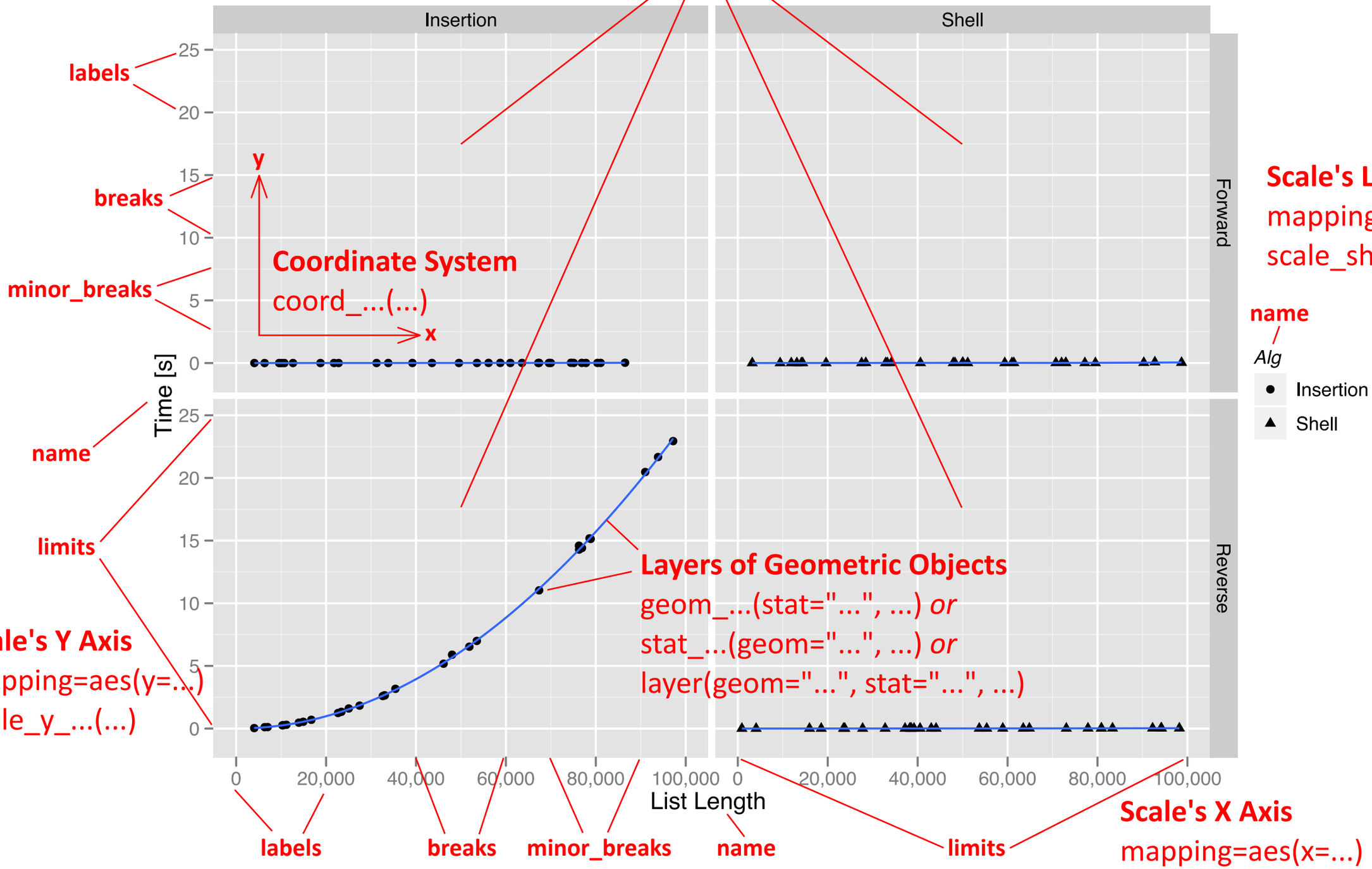
y
x
Coordinate System
coord_...(...)

Layers of Geometric Objects
geom_...(stat="...", ...) *or*
stat_...(geom="...", ...) *or*
layer(geom="...", stat="...", ...)

Scale's Y Axis
mapping=aes(y=...)
scale_y_...(...)

Scale's X Axis
mapping=aes(x=...)
scale_x_...(...)

Scale's Legend
mapping=aes(shape=...)
scale_shape_...(...)



Grammar of Graphics

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

Gr ar of Graphics

- cartesian
- flip
- trans
- equal
- polar
- map

plot ::= **coord** scale+ facet? layer+

layer ::= data mapping stat geom position?

Gram r of Graphics

x
y
size
shape
color
alpha

plot ::= coord **scale**+ facet? layer+

layer ::= data mapping stat geom position?

Grammar of Graphics

grid

wrap

plot ::= coord scale+ **facet**? layer+

layer ::= data mapping stat geom position?

Grammar of Graphics

plot ::= coord scale+ facet? layer+

layer ::= data mapping **stat** geom position?

- identity
- bin
- boxplot
- countour
- density
- smooth

Grammar of Graphics

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat **geom** position?

- point
- rug
- jitter
- boxplot
- step
- text

Grammar of Graphics

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom **position**?

- identity
- jitter
- dodge
- stack
- fill

Grammar of Graphics

plot ::= coord scale+ facet? layer+

layer ::= data mapping stat geom position?

http://sape.inf.usi.ch/quick-reference/ggplot2

Exercise

Create 4 plots with ggplot2
to **clearly** show the four phenomena
in the following data.

Save to PDF with pdf(...), ggplot(...)..., dev.off()

Present your plots in class.

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)
```

```
d$time = rexp(nrow(d), 0.01)+1000
```

1

```
d$time =
d$time + abs(d$heapSize-3)*100
```

2

```
d$time[d$opt=='on'] =
d$time[d$opt=='on'] - 200
```

3

```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4

```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)
```

Creates a data.frame, d,
with 3 variables (gc, opt, heapSize)
affecting performance,
for 11 different benchmarks,
with 10 observations (obs) for each configuration.

1

2

3

```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4

```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)
```

The data.frame d will have

length(obs)*length(Benchmark)*length(gc)*length(opt)*length(heapSize)

=

10*11*5*12*6

=

6600 rows

4

```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)
```

```
d$time = rexp(nrow(d), 0.01)+1000
```

Add a time column to the data.frame,
fill it with exponentially-distributed random values.

This data now looks like coming from an experiment.
But none of the variables have any influence on time
(time is completely random).

Data Frame

```
d = expand.grid(
```

Now let's modify time a bit,
so that its values depend on the various variables.

We perform 4 modifications, corresponding to 4 somewhat realistic phenomena.

1
```
d$time =
d$time + abs(d$heapSize-3)*100
```

2
```
d$time[d$opt=='on'] =
d$time[d$opt=='on'] - 200
```

3
```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4
```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

These 4 phenomena affected the data (e.g., heap size affects time).

Your task: Create 4 plots with ggplot2, each focusing on one phenomenon.

```
d$time = rexp(nrow(d), 0.01)+1000
```

1
```
d$time =
d$time + abs(d$heapSize-3)*100
```

2
```
d$time[d$opt=='on'] =
d$time[d$opt=='on'] - 200
```

3
```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4
```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)

d$time = rexp(nrow(d), 0.01)+1000
```

Note:
heapSize is **not** a factor,
but you can use
factor(heapSize) if you
want categorical data

1
```
d$time =
d$time + abs(d$heapSize-3)*100
```

2
```
d$time[d$opt=='on'] =
d$time[d$opt=='on'] - 200
```

3
```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4
```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```

Data Frame

```
d = expand.grid(
  obs       = 1:10,
  benchmark = c('antlr', 'bloat', 'chart', 'eclipse', 'fop', 'hsqldb',
                'jython', 'luindex', 'lusearch', 'pmd', 'xalan'),
  gc        = c('CopyMS', 'GenCopy', 'GenImmix', 'GenMS', 'Immix'),
  opt       = c('on', 'off'),
  heapSize = seq(from=1.5, to=4, by=0.5)
)
```

```
d$time = rexp(nrow(d), 0.01)+1000
```

1

```
d$time =
d$time + abs(d$heapSize-3)*100
```

2

```
d$time[d$opt=='on'] =
d$time[d$opt=='on'] - 200
```

3

```
d$time[d$opt=='on' & d$benchmark=='bloat'] =
d$time[d$opt=='on' & d$benchmark=='bloat'] + 190
```

4

```
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] =
d$time[d$opt=='on' & d$benchmark=='pmd' & d$gc=='Immix'] + 600
```